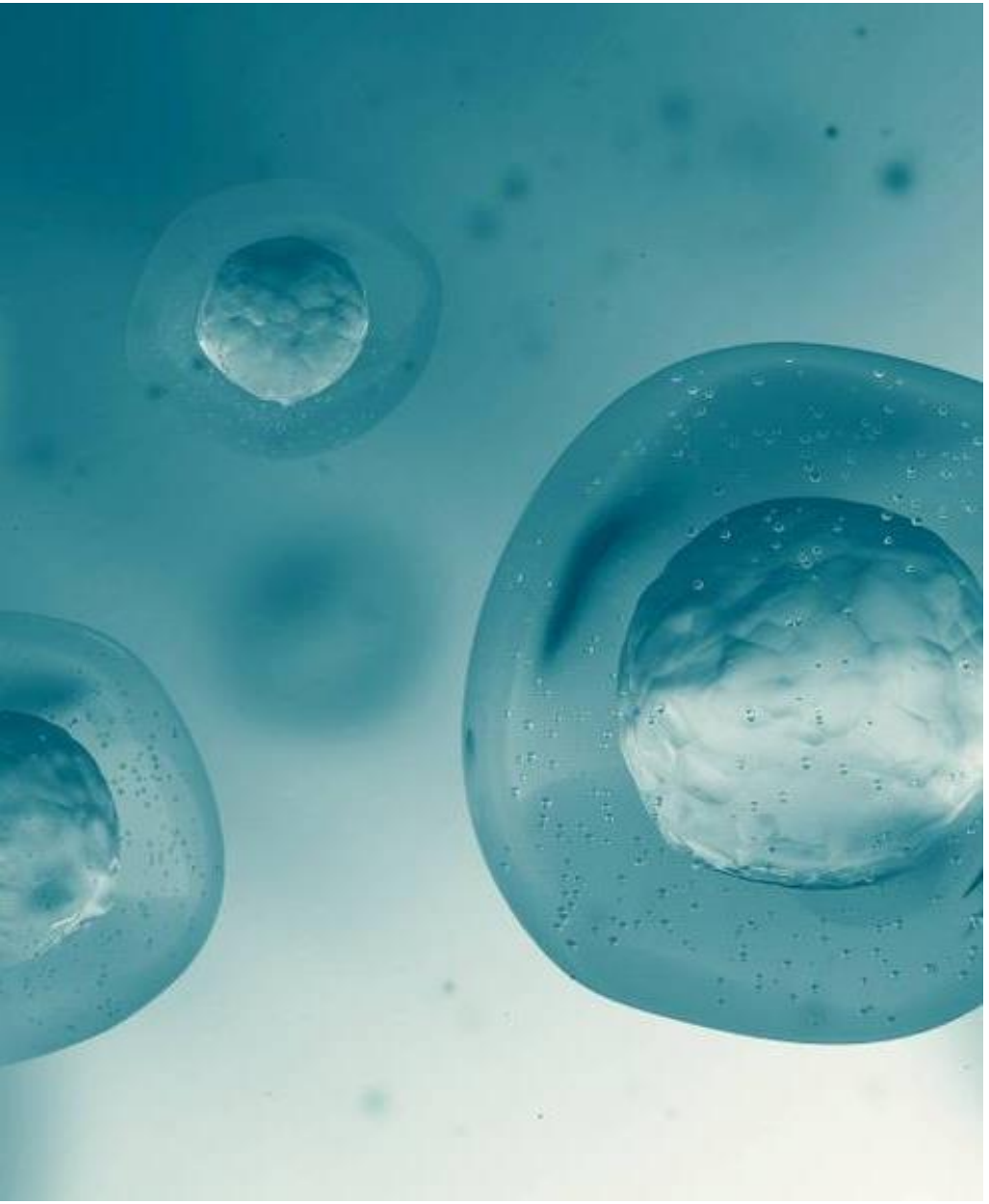
MEDICAL INSURANCE PREMIUM PREDICTION

M.Sc. Statistics

Chavi



ABSTRACT:

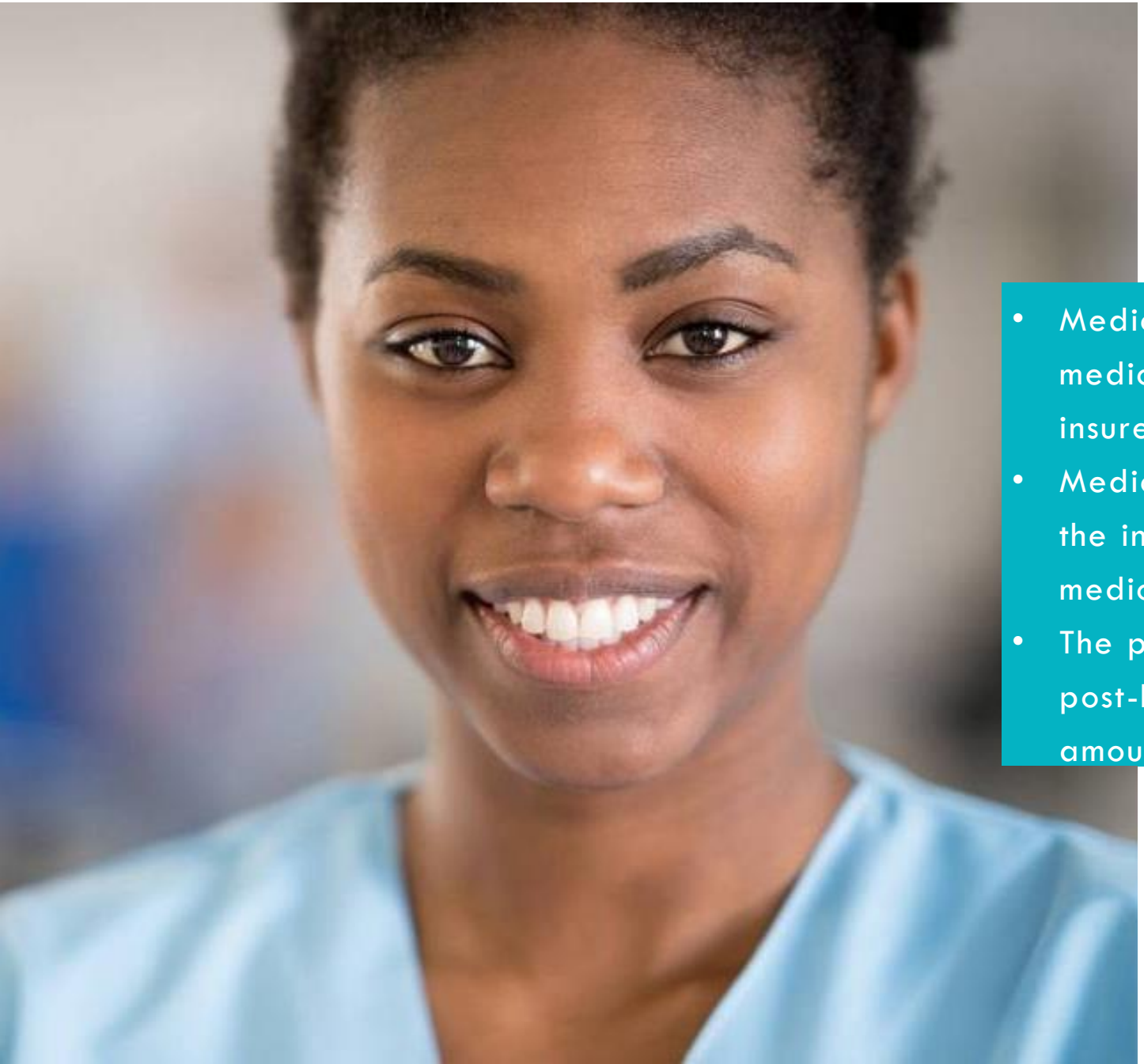
- The study explores the use of machine learning techniques to forecast medical insurance premiums, aiming to predict the most expensive expenses for patients.

- Using a comprehensive dataset, the research employs various machine learning algorithms to analyze demographic information, medical history, lifestyle factors, and insurance coverage details.

OBJECTIVE:

The objective of the project is to build a Machine Learning model to forecast annual medical insurance premium based on given customer data.

CONTENT

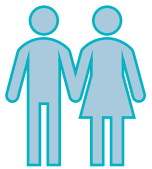


MEDICAL INSURANCE

- Medical insurance is an insurance product that covers medical and surgical expenses of whole or part of an insured individual.
- Medical insurance is a contract between the insurer and the insured. It offers coverage for expenses against medical and surgical treatments to the policyholder.
- The policy covers your in-hospitalization and pre-and post-hospitalization expenses up to a pre-determined amount.

TYPES OF MEDICAL INSURANCE:

INDIVIDUAL HEALTH INSURANCE



FAMILY FLOAT HEALTH INSURANCE

SENIOR CITIZEN HEALTH INSURANCE

CRITICAL ILLNESS INSURANCE

GROUP INSURANCE

BENEFITS:

PROTECTION

Protection against unexpected medical expenses

ACCESS

Access to quality health care service

COST SAVINGS

Tax benefits

INCENTIVE

Incentive for maintaining healthy lifestyle.

LITERATURE REVIEW:

A. KHALFAN , HASSAN & M.S. ANSARI

"Machine Learning Techniques For Predicting Insurance Premiums"
By A. Khalfan , Hassan and M.S Ansari

Link:https://www.researchgate.net/publication/380723716_MEDICAL_INSURANCE_PREMIUM_PREDICTION_WITH_MACHINE_LEARNING

RESEARCH

Machine Learning Techniques For Predicting Insurance Premiums

ABSTRACT

This study investigates various machine learning models for predicting insurance premiums, Gradient Boosting and Neural Networks. It discusses feature selection methods and model evaluation techniques to optimize prediction accuracy

R. S. RAJ & S. KUMAR

"Deep Learning Approaches for Health Insurance Premium Prediction" by R. S. Raj and S. Kumar

https://ijsrset.com/paper/11908.pdf

RESEARCH

Deep Learning Approaches for Health Insurance Premium Prediction

ABSTRACT

This paper explores the application of deep learning techniques, including Convolution Neural Network and Recurrent Neural Networks, for health insurance premium prediction. It discusses of using deep learning models in this context.

S. GUPTA & S. SHARMA

"Predicting Health Insurance Premiums : A Comparatives Study of Machine Learning Techniques" by S. Gupta and S. Sharma
https://repo.ijiert.org/index.php/ijiert/article/view/3891/3288

RESEARCH

Predicting Health Insurance Premiums: A Comparative Study of Machine Learning Techniques

ABSTRACT

This research compares the effectiveness of machine learning algorithms such KNN, Naïve Bayes , and Ensemble methods in predicting health insurance premiums. It analyzes the impact of different features sets and preprocessing techniques on prediction performance.

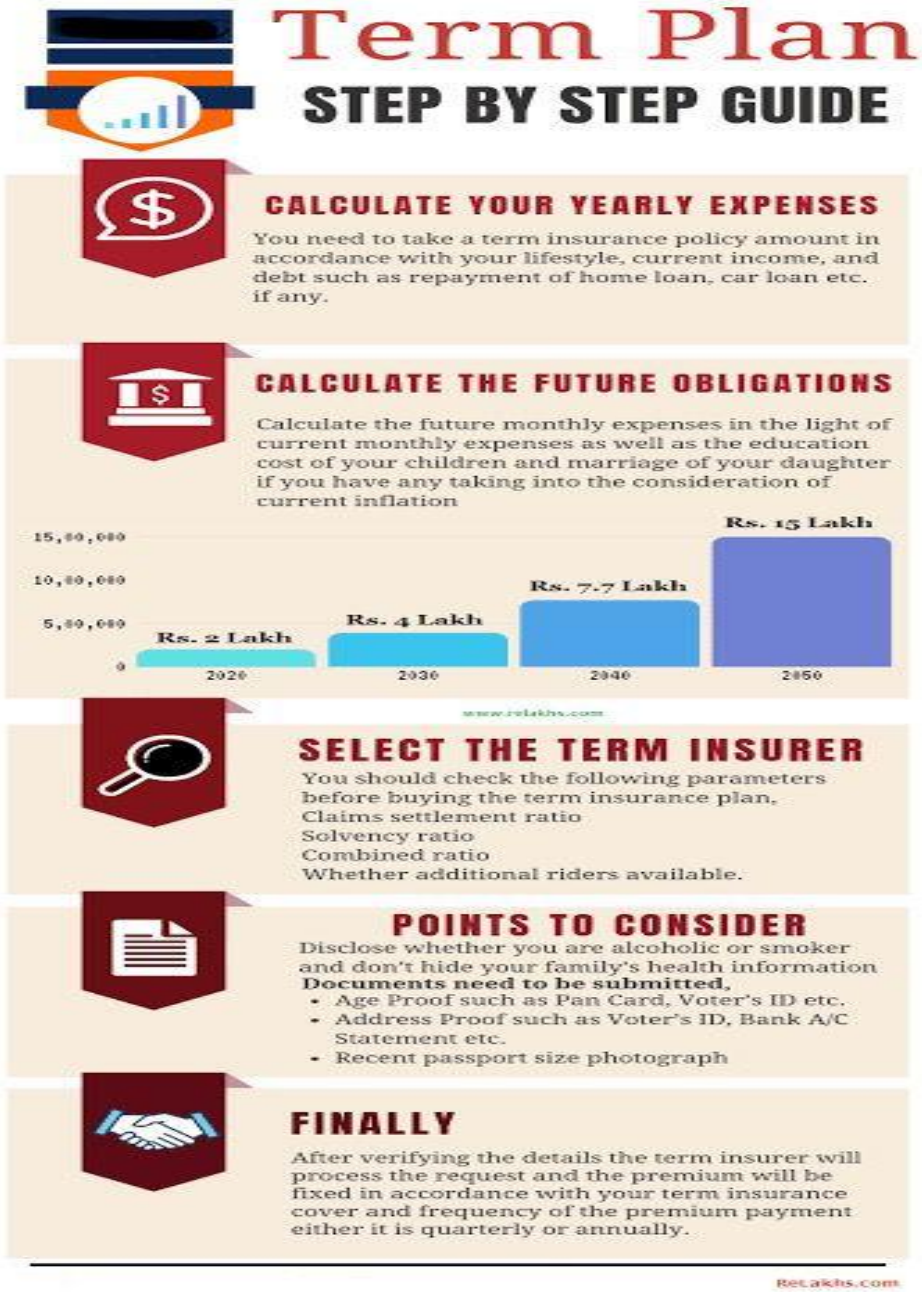
HISTORY & BACKGROUND:

- Medical insurance was launched in 1986, the medical insurance industry has grown significantly mainly due to the liberalization of the economy and general awareness. According to the World Bank, by 2010, more than 25% of India's population had access to some form of medical insurance.

- Health insurance in India is a growing segment of India's economy. The Indian healthcare system is one of the largest in the world, with the number of people it concerns: nearly 1.3 billion potential beneficiaries. The healthcare industry in India has rapidly become one of the most important sectors in the country in terms of income and job creation. In 2018, one hundred million Indian households (500 million people) benefited from health coverage. In 2011, 3.9% of India's gross domestic product was spent in the health sector.

- Policies are available that offer both individual and family coverage. Out of this 3.9%, health insurance accounts for 5-10% of expenditure, employers account for around 9% while personal expenditure amounts to an astounding 82%. In the year 2016, the NSSO released the report "Key Indicators of Social Consumption in India: Health" based on its 71st round of surveys. The survey carried out in the year 2014 found that more than 80% of Indians are not covered under any health insurance plan, and only 18% (government-funded 12%) of the urban population and 14% (government-funded 13%) of the rural population were covered under any form of health insurance.

- There are standalone health insurers along with government-sponsored health insurance providers. Recently, to improve awareness and reduce procrastination in buying health insurance, the General Insurance Corporation of India and the Insurance Regulatory and Development Authority (IRDAI) launched an awareness campaign for all segments of the population.



DATA PREPROCESSING:

DATASET OVERVIEW

age	sex	bmi	children	smoker	region	charges
19	female	27.9	0	yes	southwest	16884.92
18	male	33.77	1	no	southeast	1725.552
28	male	33	3	no	southeast	4449.462
33	male	22.705	0	no	northwest	21984.47
32	male	28.88	0	no	northwest	3866.855
31	female	25.74	0	no	southeast	3756.622
46	female	33.44	1	no	southeast	8240.59
37	female	27.74	3	no	northwest	7281.506
37	male	29.83	2	no	northeast	6406.411
60	female	25.84	0	no	northwest	28923.14
25	male	26.22	0	no	northeast	2721.321
62	female	26.29	0	yes	southeast	27808.73
23	male	34.4	0	no	southwest	1826.843
56	female	39.82	0	no	southeast	11090.72
27	male	42.13	0	yes	southeast	39611.76
19	male	24.6	1	no	southwest	1837.237
52	female	30.78	1	no	northeast	10797.34
23	male	23.845	0	no	northeast	2395.172
56	male	40.3	0	no	southwest	10602.39
30	male	35.3	0	yes	southwest	36837.47
60	female	36.005	0	no	northeast	13228.85
30	female	32.4	1	no	southwest	4149.736
18	male	34.1	0	no	southeast	1137.011
34	female	31.92	1	yes	northeast	37701.88
37	male	28.025	2	no	northwest	6203.902
59	female	27.72	3	no	southeast	14001.13
63	female	23.085	0	no	northeast	14451.84
55	female	32.775	2	no	northwest	12268.63

SOURCE:

The dataset (secondary data) was extracted from Kaggle.

https://www.kaggle.com/datasets/noordeen/insurance-premium-prediction

ABOUT:

ACME Insurance Inc. offers affordable health insurance to thousands of customer all over the United States. You're tasked with creating an automated system to estimate the annual medical expenditure for new customers, using information such as their age, sex, BMI, children, smoking habits and region of residence. Estimates from your system will be used to determine the annual insurance premium (amount paid every month) offered to the customer.

VARIABLES:

The dataset has 7 variables, namely, Age, Sex, BMI, Children, Smoker, Region and Charges. We have the data of 1338 individuals who have taken medical insurance.

DATA PREPROCESSING:

	age	bmi	children	charges	sex_male	smoker_yes	region_northwest	region_southeast	region_southwest
0	19	27.900	0	16884.92400	0	1	0	0	1
1	18	33.770	1	1725.55230	1	0	0	1	0
2	28	33.000	3	4449.46200	1	0	0	1	0
3	33	22.705	0	21984.47061	1	0	1	0	0
4	32	28.880	0	3866.85520	1	0	1	0	0

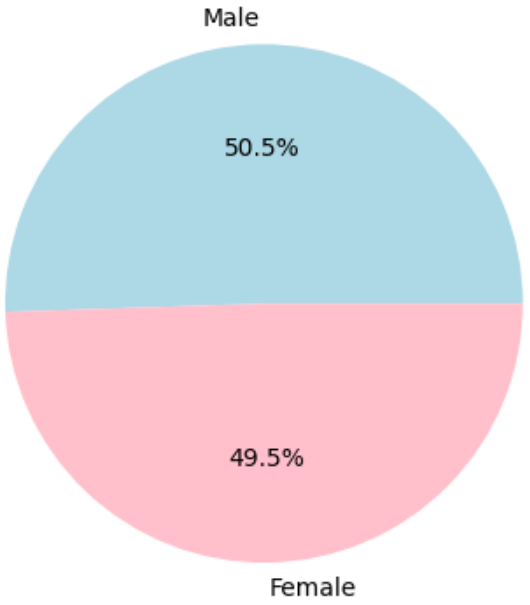
age	0
sex	0
bmi	0
children	0
smoker	0
region	0
charges	0

- The categorical variables namely sex, smoker and region were converted into numerical form using dummy variables.
- Under dummy variables, the region variable was converted into 3 variables based on the dataset.
- The dataset was checked for missing values and was observed to have no missing values.

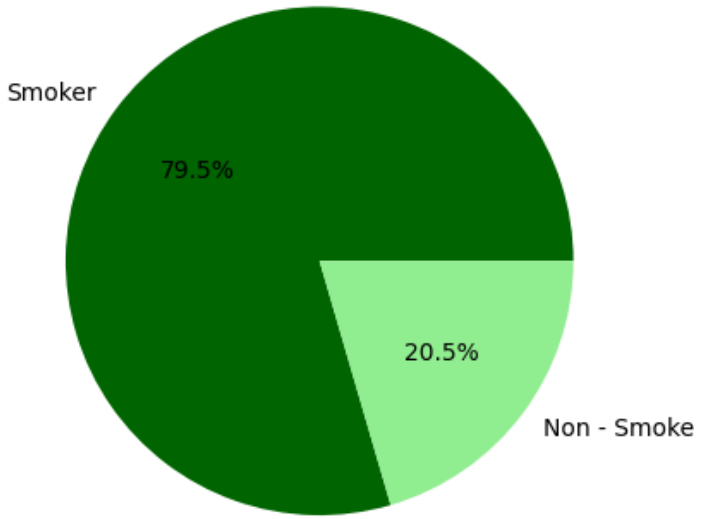
Refer to Appendix page 45

EXPLORATORY DATA ANALYSIS

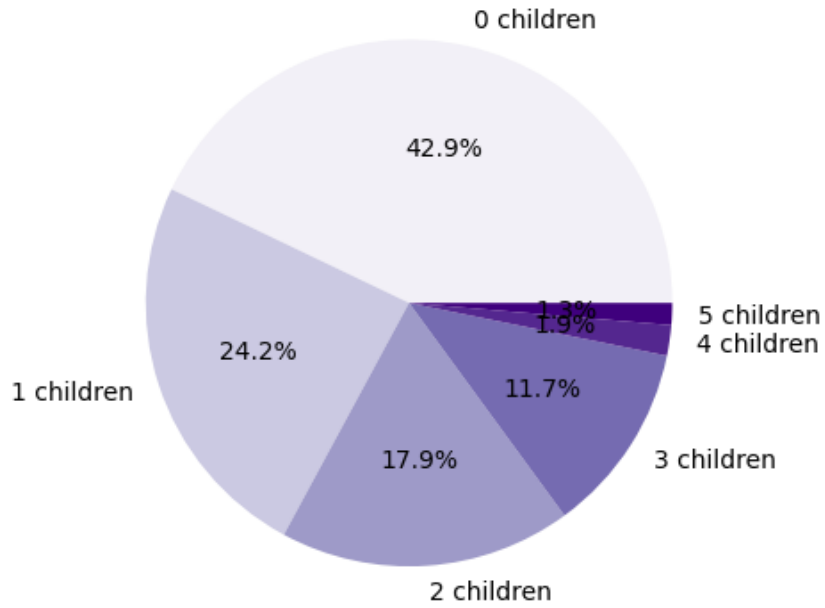
PIE CHARTS:



The sex ratio in the dataset is almost equal

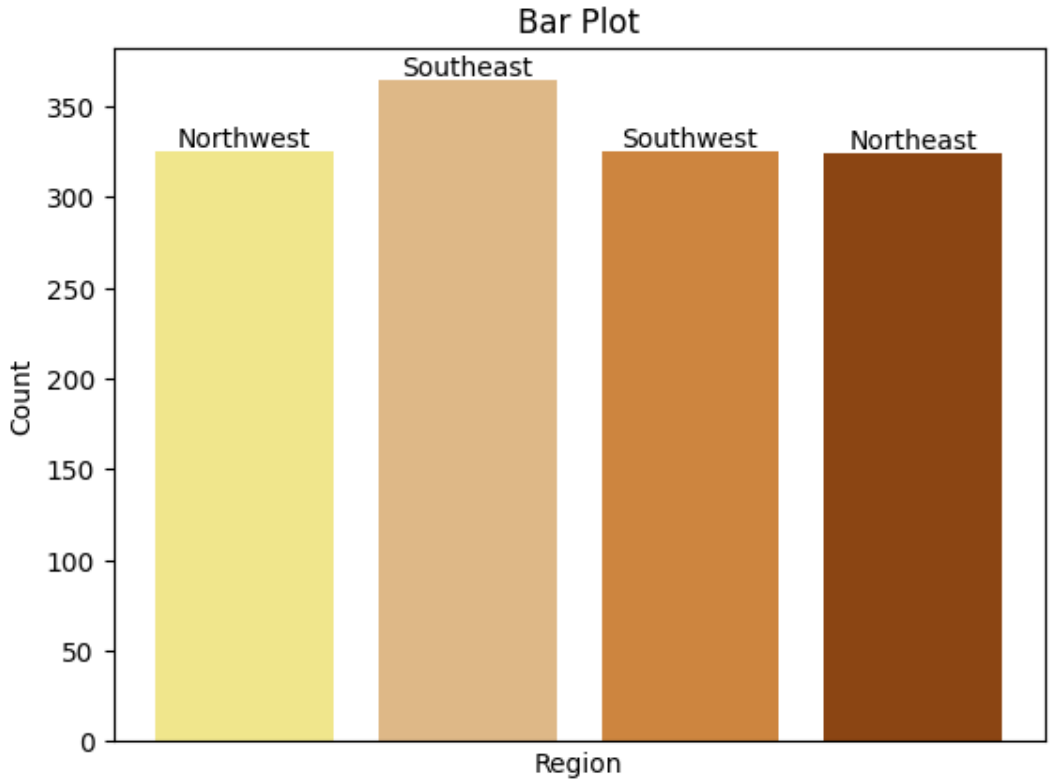


The ratio of smokers is higher than the ratio of non – smokers in the dataset

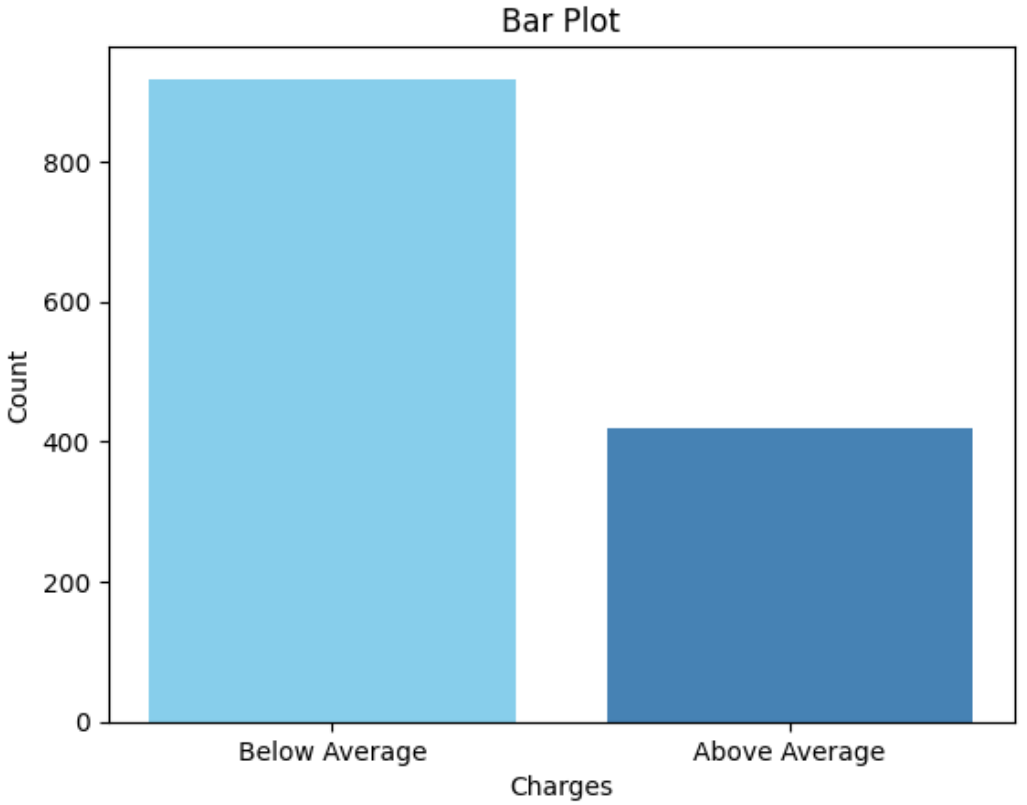


The dataset shows that around 85% of the insured are having less than 3 children

BAR PLOTS:

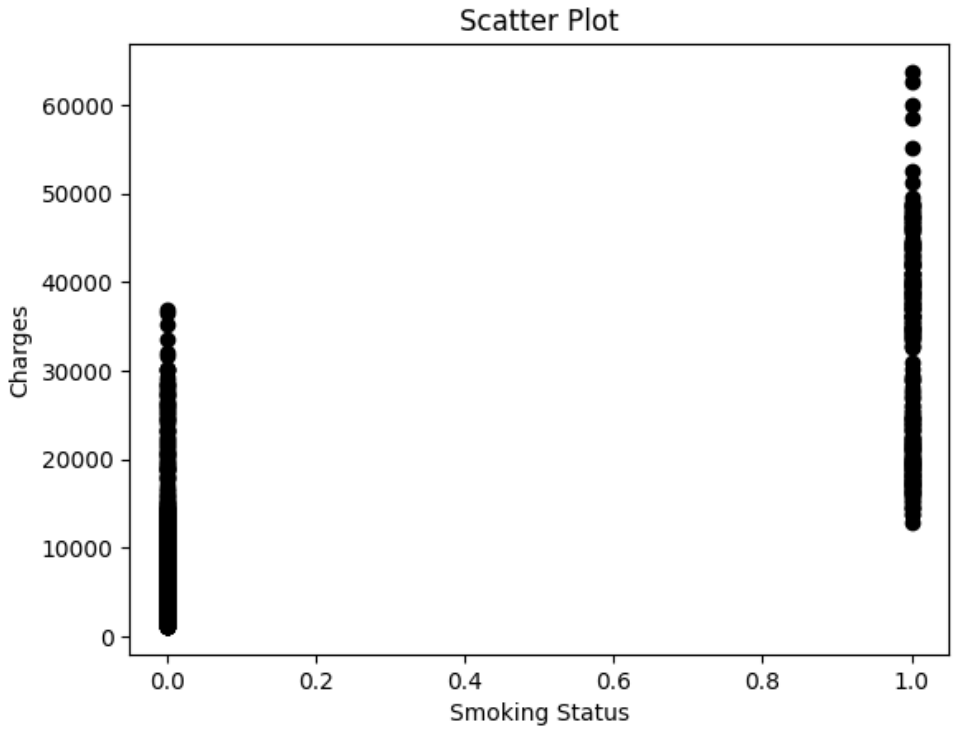


The insured belonging to the Southeast region is slightly higher than the other regions in the dataset.

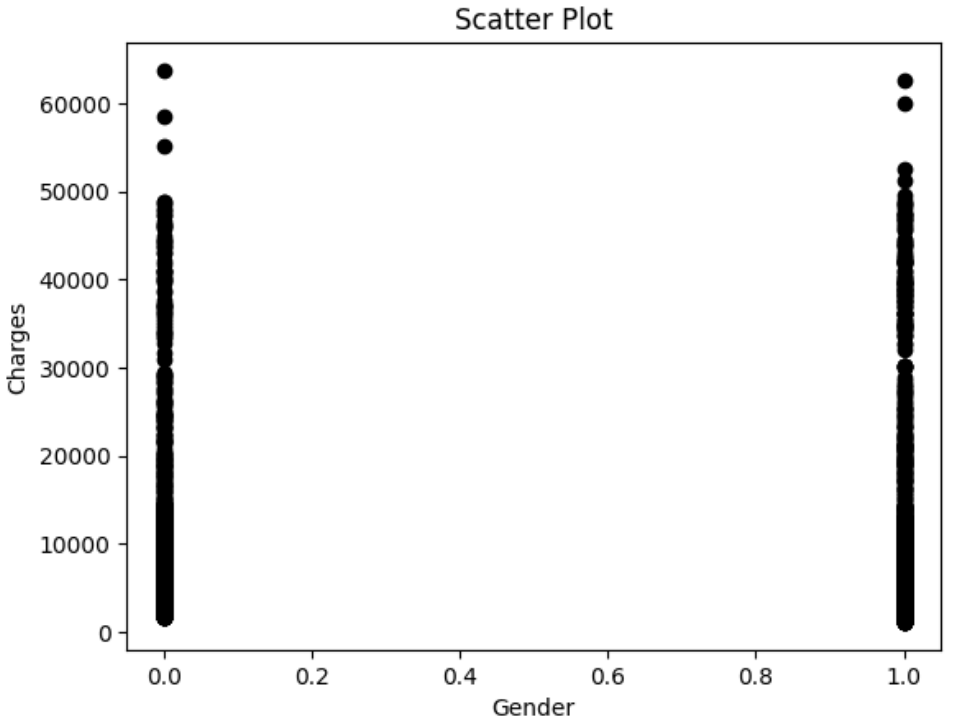


The insured being charged below average premium is more than that of the insured being charged above average premium depicting positively skewed data.

SCATTER PLOTS:



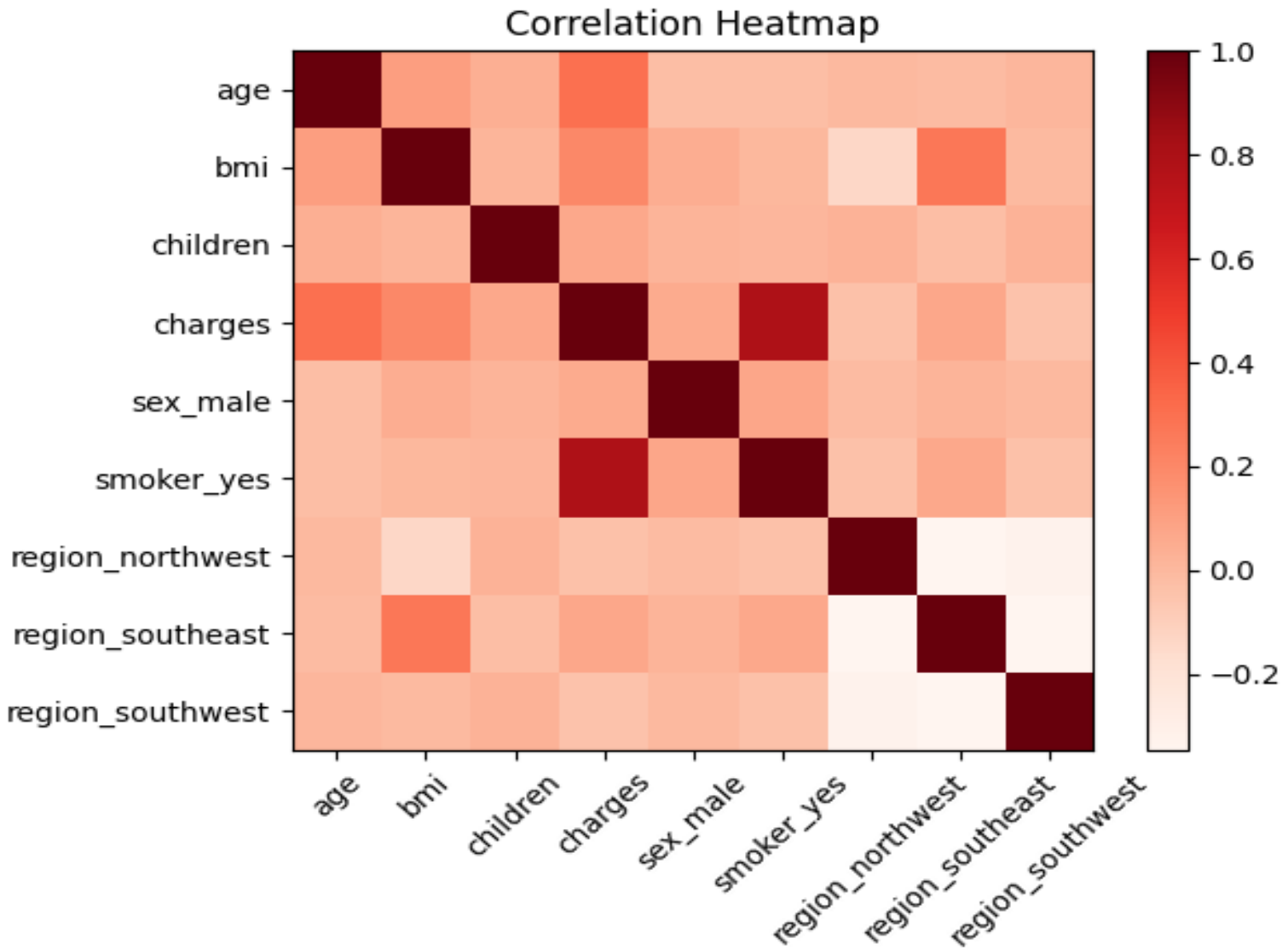
The premium charged for smokers is higher than the premium charged for non–smokers.



The premium charged for males and females are almost equal showing that premium charged is not affected by gender.

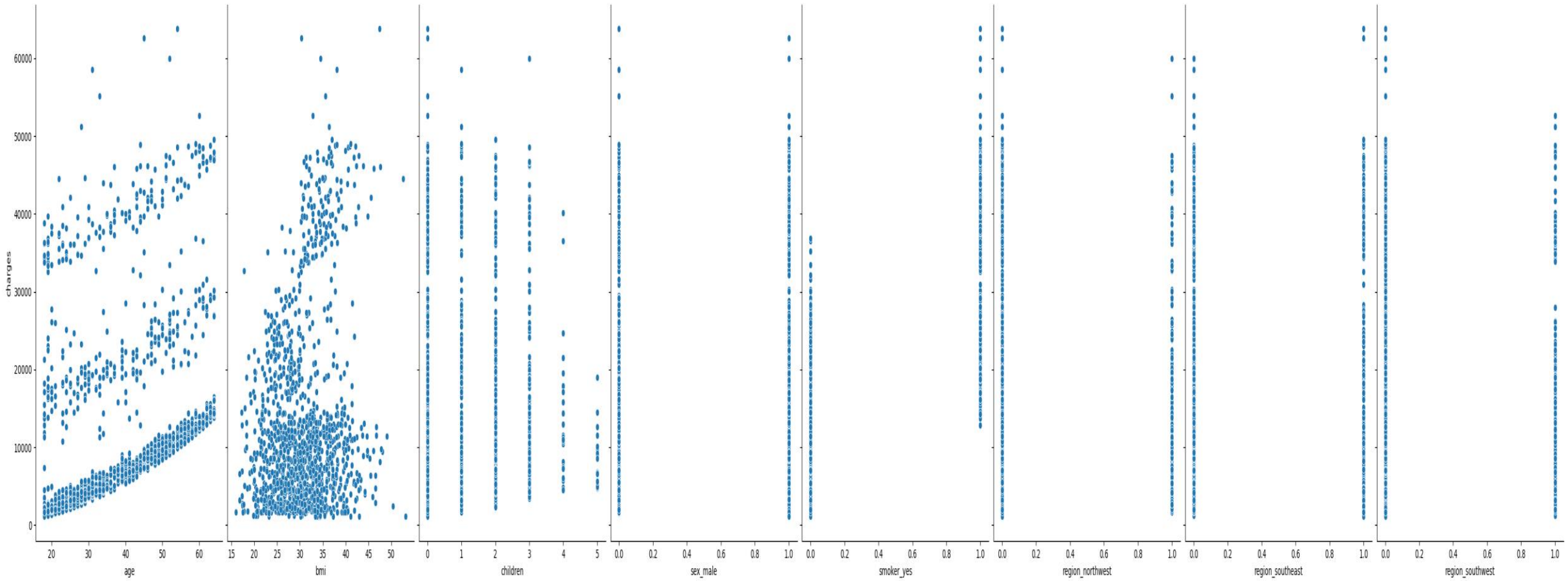
CORRELATION MATRIX

There is high positive correlation between charges and smoker_yes variables.



PAIR PLOT:

The pair plot shows charges on the y-axis and all the other variables on the x-axis.



MULTICOLLINEARITY & SIGNIFICANCE CHECK:

MULTICOLLINEARITY CHECK:

	variables	VIF
0	age	7.6
1	bmi	11.4
2	children	1.8
3	sex_male	1.9
4	smoker_yes	1.3
5	region_northwest	1.9
6	region_southeast	2.3
7	region_southwest	2.0

BMI VIF score is greater than 10.

	variables	VIF
0	age	3.8
1	children	1.8
2	sex_male	1.8
3	smoker_yes	1.2
4	region_northwest	1.7
5	region_southeast	1.8
6	region_southwest	1.7

All the VIF scores are below 10

	variables	VIF
0	age	7.5
1	bmi	7.9
2	children	1.8
3	smoker_yes	1.2

All the VIF scores are below 10 after removing insignificance

Refer to Appendix page 48

MODEL 1 (Complete Dataset)

The variables sex_male, region_northwest, region_southeast and region_southwest have p value greater than 5% for the complete dataset.

OLS Regression Results

Dep. Variable:	charges	R-squared:	0.737
Model:	OLS	Adj. R-squared:	0.735
Method:	Least Squares	F-statistic:	371.7
Date:	Thu, 08 Aug 2024	Prob (F-statistic):	1.85e-301
Time:	15:11:06	Log-Likelihood:	-10851.
No. Observations:	1070	AIC:	2.172e+04
Df Residuals:	1061	BIC:	2.177e+04
Df Model:	8		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-1.183e+04	1127.759	-10.488	0.000	-1.4e+04	-9615.183
age	253.7005	13.530	18.751	0.000	227.152	280.249
bmi	335.9628	32.228	10.424	0.000	272.724	399.201
children	436.9101	156.584	2.790	0.005	129.661	744.159
sex_male	-15.4637	378.193	-0.041	0.967	-757.555	726.627
smoker_yes	2.361e+04	470.606	50.159	0.000	2.27e+04	2.45e+04
region_northwest	-260.1327	550.305	-0.473	0.637	-1339.943	819.678
region_southeast	-913.2788	549.905	-1.661	0.097	-1992.304	165.747
region_southwest	-761.9487	543.309	-1.402	0.161	-1828.031	304.134

Omnibus:	256.825	Durbin-Watson:	1.994
Prob(Omnibus):	0.000	Jarque-Bera (JB):	620.044
Skew:	1.279	Prob(JB):	2.29e-135
Kurtosis:	5.715	Cond. No.	314.

SIGNIFICANCE TEST

MODEL 2 (Data after removing Multicollinearity)

The variables sex_male, region_northwest, region_southeast and region_southwest have p-value greater than 5%. The multicollinear variable has been removed and the R squared value has dropped.

OLS Regression Results

Dep. Variable:	charges	R-squared:	0.710
Model:	OLS	Adj. R-squared:	0.708
Method:	Least Squares	F-statistic:	371.6
Date:	Thu, 08 Aug 2024	Prob (F-statistic):	2.42e-280
Time:	15:11:06	Log-Likelihood:	-10904.
No. Observations:	1070	AIC:	2.182e+04
Df Residuals:	1062	BIC:	2.186e+04
Df Model:	7		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-2696.2870	745.394	-3.617	0.000	-4158.899	-1233.675
age	270.2830	14.101	19.168	0.000	242.614	297.952
children	466.2764	164.303	2.838	0.005	143.881	788.672
sex_male	102.6243	396.723	0.259	0.796	-675.826	881.075
smoker_yes	2.343e+04	493.580	47.474	0.000	2.25e+04	2.44e+04
region_northwest	-327.8673	577.488	-0.568	0.570	-1461.013	805.279
region_southeast	553.1110	557.907	0.991	0.322	-541.614	1647.836
region_southwest	-361.8117	568.761	-0.636	0.525	-1477.834	754.211

Omnibus:	234.326	Durbin-Watson:	2.027
Prob(Omnibus):	0.000	Jarque-Bera (JB):	576.392
Skew:	1.160	Prob(JB):	6.89e-126
Kurtosis:	5.746	Cond. No.	202.

SIGNIFICANCE TEST

MODEL 3 (Data after removing Insignificant Variables)

All the variables are having p-value below 5% including the BMI variable.

OLS Regression Results

Dep. Variable:	charges	R-squared:	0.736
Model:	OLS	Adj. R-squared:	0.735
Method:	Least Squares	F-statistic:	742.8
Date:	Thu, 08 Aug 2024	Prob (F-statistic):	3.00e-306
Time:	15:11:06	Log-Likelihood:	-10853.
No. Observations:	1070	AIC:	2.172e+04
Df Residuals:	1065	BIC:	2.174e+04
Df Model:	4		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-1.191e+04	1059.711	-11.239	0.000	-1.4e+04	-9831.179
age	254.9715	13.506	18.878	0.000	228.470	281.473
bmi	320.6190	30.679	10.451	0.000	260.422	380.816
children	430.5457	156.103	2.758	0.006	124.242	736.849
smoker_yes	2.359e+04	467.981	50.403	0.000	2.27e+04	2.45e+04

Omnibus:	257.442	Durbin-Watson:	1.993
Prob(Omnibus):	0.000	Jarque-Bera (JB):	620.986
Skew:	1.282	Prob(JB):	1.43e-135
Kurtosis:	5.712	Cond. No.	289.

SIGNIFICANCE TEST

MODEL 4 (Data after removing Multicollinearity & Insignificant Variables)

All variables are having p value below 5%

OLS Regression Results

Dep. Variable:	charges	R-squared:	0.709
Model:	OLS	Adj. R-squared:	0.708
Method:	Least Squares	F-statistic:	866.1
Date:	Thu, 08 Aug 2024	Prob (F-statistic):	3.43e-285
Time:	15:11:06	Log-Likelihood:	-10905.
No. Observations:	1070	AIC:	2.182e+04
Df Residuals:	1066	BIC:	2.184e+04
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-2678.3764	614.334	-4.360	0.000	-3883.818	-1472.935
age	270.5496	14.089	19.204	0.000	242.905	298.194
children	454.0865	163.818	2.772	0.006	132.644	775.529
smoker_yes	2.352e+04	491.110	47.884	0.000	2.26e+04	2.45e+04

Omnibus:	234.634	Durbin-Watson:	2.021
Prob(Omnibus):	0.000	Jarque-Bera (JB):	576.434
Skew:	1.162	Prob(JB):	6.74e-126
Kurtosis:	5.743	Cond. No.	133.

Refer to Appendix pg. 49

SIGNIFICANCE TEST

MACHINE LEARNING ALGORITHMS

ANALYSIS:

MODELS	Meaning
Model 1	Complete dataset
Model 2	Data after removing Multicollinearity
Model 3	Data after removing Insignificant Variables
Model 4	Data after removing Multicollinearity & Insignificant Variables

SPLITTING RATIOS (TRAIN : TEST)
60:40, 70:30, 75:25, 80:20, 90:10

MACHINE LEARNING ALGORITHMS:	
Multiple Linear Regression	Decision Tree Regressor
K Nearest Neighbor	Random Forest Regressor
Support Vector Machine (Regressor)	Bagging on Decision Tree Regressor

60 : 40 TRAIN - TEST SPLIT RATIO

ML Algorithm:	Model 1	Model 2	Model 3	Model 4
Linear Regression	0.74 6135.87	0.70 6524.69	0.74 6130.19	0.70 6514.23
KNN	0.07 11582.53	0.31 9938.22	0.02 11845.11	0.48 8686.53
SVM Regressor	-0.17 12960.35	-0.167 12952.64	-0.169 12963.70	-0.167 12956.69
Decision Tree Regressor	0.66 7041.94	0.38 9426.84	0.67 6929.59	0.51 8417.39
Random Forest Regressor	0.83 4980.66	0.62 7387.52	0.82 5116.41	0.60 7608.54
Bagging	0.82 5154.70	0.61 7510.15	0.83 4928.13	0.60 7574.18

- **Model 1 – Complete dataset**
- **Model 2 – Removing Multicollinearity**
- **Model 3 – Removing Insignificant Variables**
- **Model 4 – Removing both Multicollinearity & Insignificant Variables**

● R Squared Values

● Root Mean Square Error Values

70 : 30 TRAIN - TEST SPLIT RATIO

ML Algorithm:	Model 1	Model 2	Model 3	Model 4
Linear Regression	0.74 6063.12	0.707 6443.53	0.74 6079.47	0.706 6455.40
KNN	0.14 11037.96	0.35 9619.63	0.13 11115.54	0.48 8554.57
SVM Regressor	-0.12 12619.18	-0.123 12646.79	-0.1227 12613.94	-0.1232 12616.30
Decision Tree Regressor	0.68 6765.89	0.35 9625.27	0.63 7248.04	0.51 8317.44
Random Forest Regressor	0.83 4922.84	0.56 7852.98	0.82 5014.81	0.59 7621.20
Bagging	0.83 4931.72	0.56 7861.64	0.83 4854.19	0.59 7596.40

- **Model 1 – Complete dataset**
- **Model 2 – Removing Multicollinearity**
- **Model 3 – Removing Insignificant Variables**
- **Model 4 – Removing both Multicollinearity & Insignificant Variables**

R Squared Values

Root Mean Square Error Values

75 : 25 TRAIN - TEST SPLIT RATIO

ML Algorithm:	Model 1	Model 2	Model 3	Model 4
Linear Regression	0.73 5982.56	0.70 6373.60	0.73 6005.93	0.699 6385.05
KNN	0.15 10677.08	0.37 9195.89	0.17 10558.89	0.47 8416.19
SVM Regressor	-0.0881 12098.66	-0.09 12100.85	-0.088 12100.75	-0.0889 12103.07
Decision Tree Regressor	0.66 6758.38	0.36 9277.69	0.66 6796.13	0.51 8114.06
Random Forest Regressor	0.82 4925.91	0.55 7777.17	0.81 5006.89	0.56 7661.84
Bagging	0.82 4946.53	0.58 7477.92	0.82 4898.67	0.57 7581.01

- **Model 1 – Complete dataset**
- **Model 2 – Removing Multicollinearity**
- **Model 3 – Removing Insignificant Variables**
- **Model 4 – Removing both Multicollinearity & Insignificant Variables**

● R Squared Values

● Root Mean Square Error Values

80 : 20 TRAIN - TEST SPLIT RATIO

ML Algorithm:	Model 1	Model 2	Model 3	Model 4
Linear Regression	0.762 5956.45	0.727 6383.51	0.761 5970.76	0.725 6400.02
KNN	0.17 11130.46	0.37 9665.30	0.18 11087.03	0.50 8659.96
SVM Regressor	-0.13 13013.09	-0.132 12997.11	-0.133 13004.88	-0.13 12989.05
Decision Tree Regressor	0.77 5916.54	0.42 9329.92	0.63 6681.99	0.54 8254.34
Random Forest Regressor	0.85 4775.07	0.60 7729.89	0.85 4791.28	0.61 7669.70
Bagging	0.84 4861.84	0.63 7416.43	0.86 4644.01	0.62 7573.42

- **Model 1 – Complete dataset**
- **Model 2 – Removing Multicollinearity**
- **Model 3 – Removing Insignificant Variables**
- **Model 4 – Removing both Multicollinearity & Insignificant Variables**

● R Squared Values

● Root Mean Square Error Values

90 : 10 TRAIN - TEST SPLIT RATIO

ML Algorithm:	Model 1	Model 2	Model 3	Model 4
Linear Regression	0.727 6565.38	0.689 7004.28	0.725 6588.75	0.687 7024.01
KNN	0.06 12167.61	0.26 10835.84	0.09 12005.97	0.45 9295.01
SVM Regressor	-0.31 14422.25	-0.31 14414.25	-0.32 14422.24	-0.32 14417.05
Decision Tree Regressor	0.74 6389.89	0.37 10003.52	0.71 6790.11	0.43 9453.64
Random Forest Regressor	0.84 5028.52	0.53 8595.99	0.83 5139.71	0.52 8674.63
Bagging	0.82 5372.01	0.56 8347.84	0.84 5088.65	0.53 8644.75

- **Model 1 – Complete dataset**
- **Model 2 – Removing Multicollinearity**
- **Model 3 – Removing Insignificant Variables**
- **Model 4 – Removing both Multicollinearity & Insignificant Variables**

R Squared Values

Root Mean Square Error Values

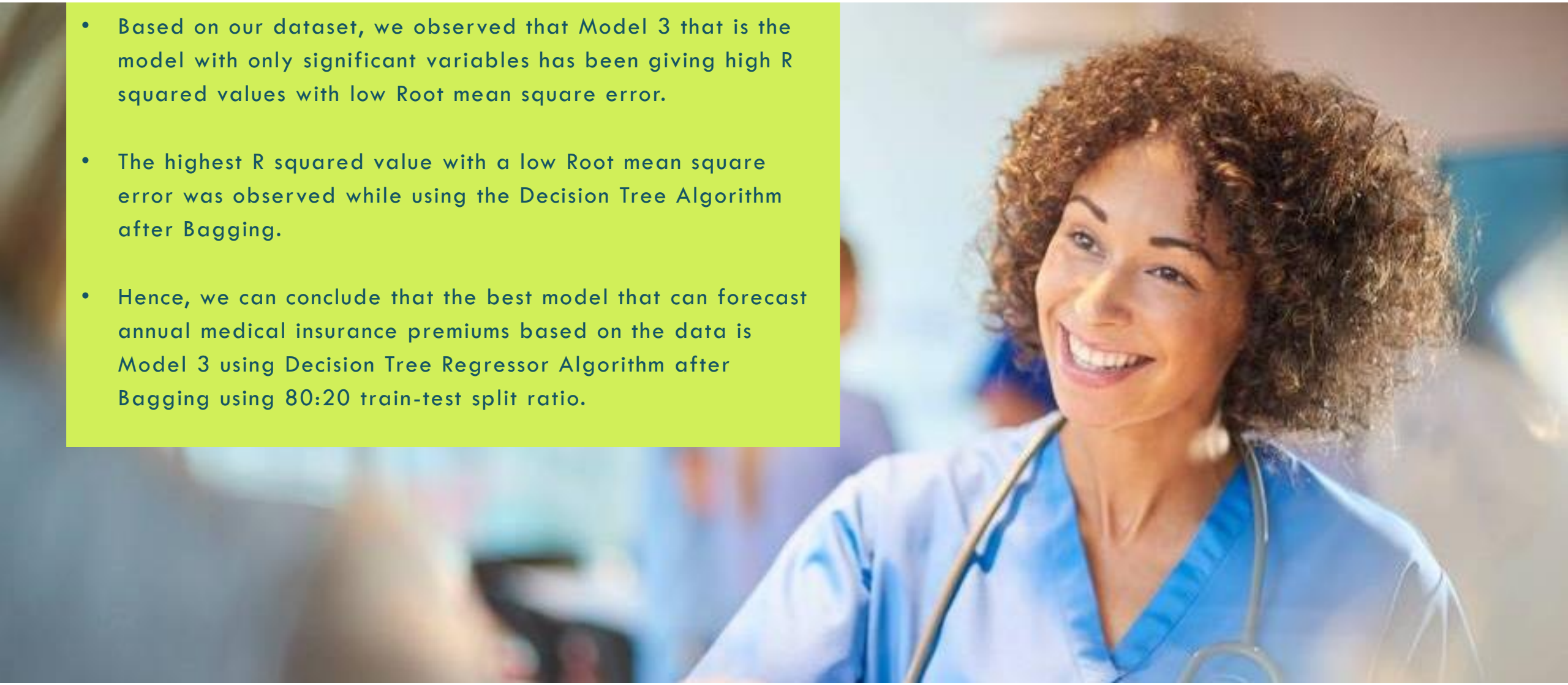
ALGORITHM COMPARISON

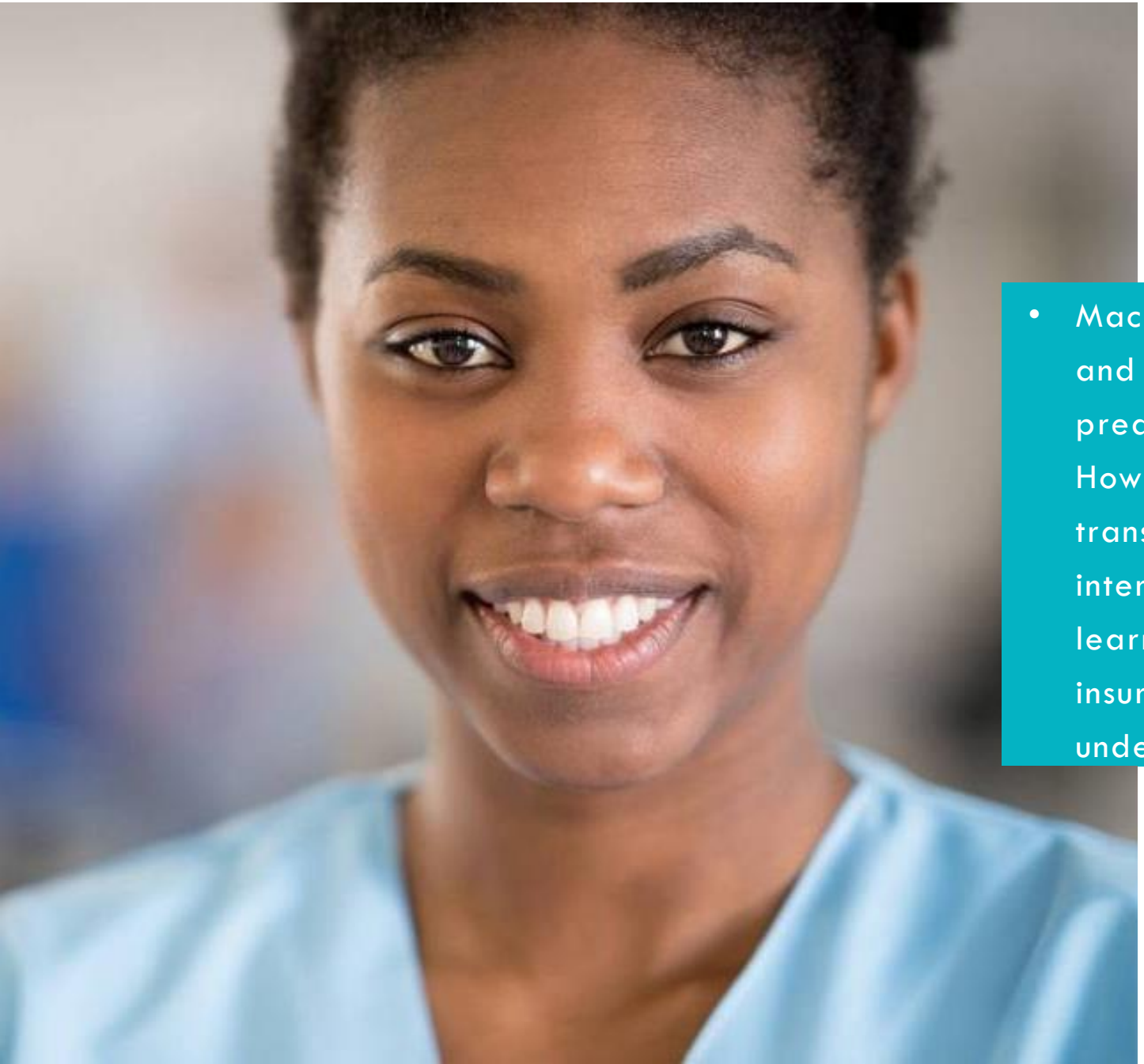
ML Algorithm:	60 : 40	70 : 30	75 : 25	80 : 20	90 : 10
Linear Regression	0.74 6130.19	0.74 6079.47	0.73 6005.93	0.761 5970.76	0.725 6588.75
KNN	0.02 11845.11	0.13 11115.54	0.17 10558.89	0.18 11087.03	0.09 12005.97
SVM Regressor	-0.169 12963.70	-0.1227 12613.94	-0.088 12100.75	-0.133 13004.88	-0.32 14422.24
Decision Tree Regressor	0.67 6929.59	0.63 7248.04	0.66 6796.13	0.63 6681.99	0.71 6790.11
Random Forest Regressor	0.82 5116.41	0.82 5014.81	0.81 5006.89	0.85 4791.28	0.83 5139.71
Bagging	0.83 4928.13	0.83 4854.19	0.82 4898.67	0.86 4644.01	0.84 5088.65

- R Squared Values
- Root Mean Square Error Values

CONCLUSION

- Based on our dataset, we observed that Model 3 that is the model with only significant variables has been giving high R squared values with low Root mean square error.
- The highest R squared value with a low Root mean square error was observed while using the Decision Tree Algorithm after Bagging.
- Hence, we can conclude that the best model that can forecast annual medical insurance premiums based on the data is Model 3 using Decision Tree Regressor Algorithm after Bagging using 80:20 train-test split ratio.





SUMMARY

- Machine learning may increase the efficiency, fairness, and accuracy of the insurance industry by creating prediction models and evaluating historical data. However, challenges include ensuring fairness, transparency, mitigating bias, and maintaining model interpretability. Despite these, advancements in machine learning techniques and a deeper understanding of insurance dynamics are paving the way for more understanding.

THANK YOU

Chavi Boob
MSc. Statistics

APPENDIX

LOADING LIBRARIES

```
[ ]  import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     import seaborn as sns
     import statsmodels.formula.api as smf
     import statsmodels.api as sm
     import xgboost as xgb
     from sklearn.linear_model import LinearRegression
     from sklearn import metrics
     from sklearn.model_selection import train_test_split
     from statsmodels.stats.outliers_influence import variance_inflation_factor
     from sklearn.neighbors import KNeighborsRegressor
     from sklearn.metrics import r2_score
     from sklearn.metrics import mean_squared_error
     from sklearn.metrics import mean_absolute_percentage_error
     from sklearn.svm import SVR
     from sklearn.tree import DecisionTreeRegressor
     from sklearn.ensemble import RandomForestRegressor
     from sklearn.ensemble import BaggingRegressor
```

LOADING DATA

```
[ ] data=pd.read_csv('/content/Insurance.csv')
    data.head()
```

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520

UNDERSTATNDING THE DATA

To get to know about the dataset:

```
[ ] data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1338 entries, 0 to 1337
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       1338 non-null   int64  
 1   sex       1338 non-null   object 
 2   bmi       1338 non-null   float64
 3   children  1338 non-null   int64  
 4   smoker    1338 non-null   object 
 5   region    1338 non-null   object 
 6   charges   1338 non-null   float64
dtypes: float64(2), int64(2), object(3)
memory usage: 73.3+ KB
```

```
data.describe()
```

	age	bmi	children	charges
count	1338.000000	1338.000000	1338.000000	1338.000000
mean	39.207025	30.663397	1.094918	13270.422265
std	14.049960	6.098187	1.205493	12110.011237
min	18.000000	15.960000	0.000000	1121.873900
25%	27.000000	26.296250	0.000000	4740.287150
50%	39.000000	30.400000	1.000000	9382.033000
75%	51.000000	34.693750	2.000000	16639.912515
max	64.000000	53.130000	5.000000	63770.428010

```
[ ] data.shape
```

```
(1338, 7)
```

Finding out unique values for required column:

The object columns -> Sex, Smoker and Region:

```
l1 = data.iloc[:,[1,4,5]]
for col_name in l1.columns:  # Iterate over column names instead of the DataFrame itself
  print(data[col_name].unique())
```

```
['female' 'male']
['yes' 'no']
['southwest' 'southeast' 'northwest' 'northeast']
```

Dummy variables encoding:

```
X = pd.get_dummies(data, columns=['sex', 'smoker', 'region'], drop_first=True)
X.head()
```

	age	bmi	children	charges	sex_male	smoker_yes	region_northwest	region_southeast	region_southwest
0	19	27.900	0	16884.92400	False	True	False	False	True
1	18	33.770	1	1725.55230	True	False	False	True	False
2	28	33.000	3	4449.46200	True	False	False	True	False
3	33	22.705	0	21984.47061	True	False	True	False	False
4	32	28.880	0	3866.85520	True	False	True	False	False

Changing the type of dummy variables by splitting the dataset:

```
X1 = X.iloc[:,[-5,-4,-3,-2,-1]]
X1 = X1.astype('int')
X1.head()
```

	sex_male	smoker_yes	region_northwest	region_southeast	region_southwest
0	0	1	0	0	1
1	1	0	0	1	0
2	1	0	0	1	0
3	1	0	1	0	0
4	1	0	1	0	0

```
X2 = X.iloc[:,[0,1,2,3]]
X2.head()
```

	age	bmi	children	charges
0	19	27.900	0	16884.92400
1	18	33.770	1	1725.55230
2	28	33.000	3	4449.46200
3	33	22.705	0	21984.47061
4	32	28.880	0	3866.85520

Joining the dataset back in required format:

```
[ ] x = X2.join(X1)
    x.head()
```

	age	bmi	children	charges	sex_male	smoker_yes	region_northwest	region_southeast	region_southwest
0	19	27.900	0	16884.92400	0	1	0	0	1
1	18	33.770	1	1725.55230	1	0	0	1	0
2	28	33.000	3	4449.46200	1	0	0	1	0
3	33	22.705	0	21984.47061	1	0	1	0	0
4	32	28.880	0	3866.85520	1	0	1	0	0

Checking the structue of the new data frame:

```
x.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1338 entries, 0 to 1337
Data columns (total 9 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   age               1338 non-null   int64  
 1   bmi               1338 non-null   float64
 2   children          1338 non-null   int64  
 3   charges           1338 non-null   float64
 4   sex_male          1338 non-null   int64  
 5   smoker_yes        1338 non-null   int64  
 6   region_northwest  1338 non-null   int64  
 7   region_southeast  1338 non-null   int64  
 8   region_southwest  1338 non-null   int64  
```

Multicoliniarity Check

```
[ ] # create X and y

    X = x.drop('charges', axis=1)
    y = x["charges"]

    # Split Data into Training and Test Sets
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

```
[ ] # VIF
    def calc_vif(X):

        # Calculating VIF
        vif = pd.DataFrame()
        vif["variables"] = X.columns
        vif["VIF"] = [variance_inflation_factor(X.values, i).round(1) for i in range(X.shape[1])]

        return(vif)

    calc_vif(X_train)
```

	variables	VIF
0	age	7.6
1	bmi	11.4
2	children	1.8
3	sex_male	1.9
4	smoker_yes	1.3
5	region_northwest	1.9
6	region_southeast	2.3

Here, the BMI is giving value greater than 10, depicting multicollinearity. Hence we will move on removing BMI variable.

```
calc_vif(X_train.drop('bmi', axis=1))
```

	variables	VIF
0	age	3.8
1	children	1.8
2	sex_male	1.8
3	smoker_yes	1.2
4	region_northwest	1.7
5	region_southeast	1.8
6	region_southwest	1.7

Now, we see that all the variables are below 10 which shows no multicollienarity.

Significance Test (Using p - values)

```
X_train_constant = sm.add_constant(X_train)
model = sm.OLS(y_train, X_train_constant).fit()

# Print the summary to get p-values
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                charges   R-squared:                       0.737
Model:                            OLS   Adj. R-squared:                  0.735
Method:                 Least Squares   F-statistic:                     371.7
Date:                Mon, 12 Aug 2024   Prob (F-statistic):          1.85e-301
Time:                        12:32:18   Log-Likelihood:                -10851.
No. Observations:                1070   AIC:                         2.172e+04
Df Residuals:                    1061   BIC:                         2.177e+04
Df Model:                           8
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const      -1.183e+04   1127.759    -10.488      0.000    -1.4e+04   -9615.183
age            253.7005     13.530     18.751      0.000     227.152     280.249
bmi            335.9628     32.228     10.424      0.000     272.724     399.201
children       436.9101    156.584      2.790      0.005     129.661     744.159
sex_male       -15.4637    378.193     -0.041      0.967    -757.555     726.627
smoker_yes    2.361e+04    470.606     50.159      0.000    2.27e+04    2.45e+04
region_northwest  -260.1327    550.305     -0.473      0.637   -1339.943     819.678
region_southeast  -913.2788    549.905     -1.661      0.097   -1992.304     165.747
region_southwest  -761.9487    543.309     -1.402      0.161   -1828.031     304.134
==============================================================================
Omnibus:                      256.825   Durbin-Watson:                   1.994
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              620.044
Skew:                           1.279   Prob(JB):                     2.29e-135
Kurtosis:                       5.715   Cond. No.                         314.
```

```
X_train_nomulti_4 = X_train_nomulti.drop(['sex_male','region_northwest','region_southwest','region_southeast'], axis=1)
X_train_constant_5 = sm.add_constant(X_train_nomulti_4)
model6 = sm.OLS(y_train, X_train_constant_5).fit()

# Print the summary to get p-values
print(model6.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                charges   R-squared:                       0.709
Model:                            OLS   Adj. R-squared:                  0.708
Method:                 Least Squares   F-statistic:                     866.1
Date:                Mon, 12 Aug 2024   Prob (F-statistic):          3.43e-285
Time:                        12:32:18   Log-Likelihood:                -10905.
No. Observations:                1070   AIC:                         2.182e+04
Df Residuals:                    1066   BIC:                         2.184e+04
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const      -2678.3764    614.334     -4.360      0.000   -3883.818   -1472.935
age            270.5496     14.089     19.204      0.000     242.905     298.194
children       454.0865    163.818      2.772      0.006     132.644     775.529
smoker_yes    2.352e+04    491.110     47.884      0.000    2.26e+04    2.45e+04
==============================================================================
Omnibus:                      234.634   Durbin-Watson:                   2.021
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              576.434
Skew:                           1.162   Prob(JB):                    6.74e-126
Kurtosis:                       5.743   Cond. No.                         133.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

LINEAR REGRESSION

```
[ ]  # create X and y
     feature = ['age','bmi','children','sex_male','smoker_yes','region_northwest','region_southeast','region_southwest']
     X_1 = x[feature]
     y_1 = x.charges

     # instantiate and fit
     lm1 = LinearRegression()
     lm1.fit(X_1, y_1)

     # print the coefficients
     print(lm1.intercept_)
     print(lm1.coef_)
```

```
-11938.538576167146
[  256.85635254   339.19345361   475.50054515  -131.3143594
 23848.53454191  -352.96389942 -1035.02204939  -960.0509913 ]
```

```
     # pair the feature names with the coefficients
     {i:j for i,j in zip(feature, lm1.coef_)}
```

```
{'age': 256.85635253734864,
 'bmi': 339.1934536108373,
 'children': 475.5005451491264,
 'sex_male': -131.31435939511314,
 'smoker_yes': 23848.534541912835,
 'region_northwest': -352.96389942465464,
 'region_southeast': -1035.022049387825,
 'region_southwest': -960.0509913008366}
```

```
[ ] X_train, X_test, y_train, y_test = train_test_split(X_1, y_1, random_state=1)
    lm = LinearRegression()
    lm.fit(X_train, y_train)
    LM_y_pred = lm.predict(X_test)
    print(np.sqrt(metrics.mean_squared_error(y_test, LM_y_pred)))
```

```
5982.5670650360735
```

```
# Different splitting ratios
split_ratios = [0.1, 0.2, 0.3, 0.4]

for ratio in split_ratios:
    Lm_X_train, Lm_X_test, Lm_y_train, Lm_y_test = train_test_split(X_1, y_1, test_size=ratio, random_state=1)
    LM1 = LinearRegression()
    LM1.fit(Lm_X_train, Lm_y_train)
    Lm_y_pred = LM1.predict(Lm_X_test)
    rmse = np.sqrt(metrics.mean_squared_error(Lm_y_test, Lm_y_pred))
    R2 = r2_score(Lm_y_test, Lm_y_pred)
    print(f"Split Ratio: {ratio}, RMSE: {rmse}, R Score : {R2}")
```

```
Split Ratio: 0.1, RMSE: 6565.384612114565, R Score : 0.7272291228241008
Split Ratio: 0.2, RMSE: 5956.454717976427, R Score : 0.7623311844057112
Split Ratio: 0.3, RMSE: 6063.122656850449, R Score : 0.7405989316927213
Split Ratio: 0.4, RMSE: 6135.876953164414, R Score : 0.7380715985194329
```

KNN ALGORITHM

```
[ ] KNN_model=KNeighborsRegressor(n_neighbors=5)
```

```
[ ] KNN_model.fit(X_train, y_train)
```

```
KNeighborsRegressor
KNeighborsRegressor()
```

```
KNN_y_pred = KNN_model.predict(X_test)
```

```
[ ] KNN_y_pred
```

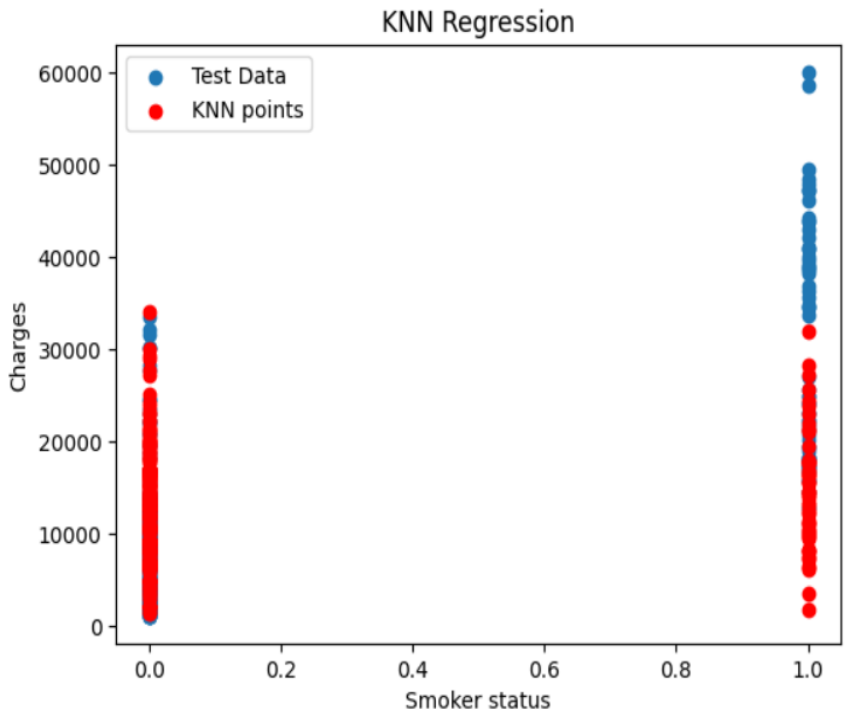
```
array([15156.85964  , 18192.71901  ,  9134.22192  , 20422.064668 ,
        4921.11803  , 21110.20065  , 12899.704722 , 14610.84098  ,
        6123.67562  ,  8260.90652  , 18282.674486 , 21414.624642 ,
       13324.41081  ,  7052.57442  ,  3348.59957  , 12664.60863  ,
        7087.627102 ,  9390.89367  , 14256.47299  , 19929.52616  ,
       16422.02406  , 14646.960684 , 12133.41331  , 12598.010172 ,
        6847.7330338,  6619.29417  , 15867.571668 , 15388.93878  ,
        6838.146094 ,  3823.70073  , 13665.119422 , 10955.58986  ,
       16897.74048  , 15424.187706 , 17891.23815  , 19428.691792 ,
       31942.651644 , 10057.42127  , 13399.448678 , 25581.588528 ,
       10005.126852 , 11983.0098   , 10505.44012  , 21049.676    ,
       10115.164202 , 12752.05873  ,  1493.93174  , 15675.30043  ,
        9090.84772  , 16109.62994  , 28940.837676 , 17779.72195  ,
        7034.659608 ,  8827.49297  ,  6638.432198 ,  5913.833034 ,
        8050.35638  , 17832.697746 ,  9060.315316 , 16702.8806   ,
       23463.51687  , 27768.940838 , 11106.923886 ,  7868.462224 ,
       14852.05197  ,  9234.9597   , 13467.79109  , 11876.53797  ,
        2233.44006  ,  6418.357988 , 24486.92674  , 14242.232154 ,
        5058.69263  , 17004.849406 , 16637.80268  ,  9705.024624 ,
        4556.85977  ,  6761.15168  , 16115.212744 , 16619.632168 ,
```

```
knn = pd.DataFrame({'Predicted':KNN_y_pred,'Actual':y_test})
knn
```

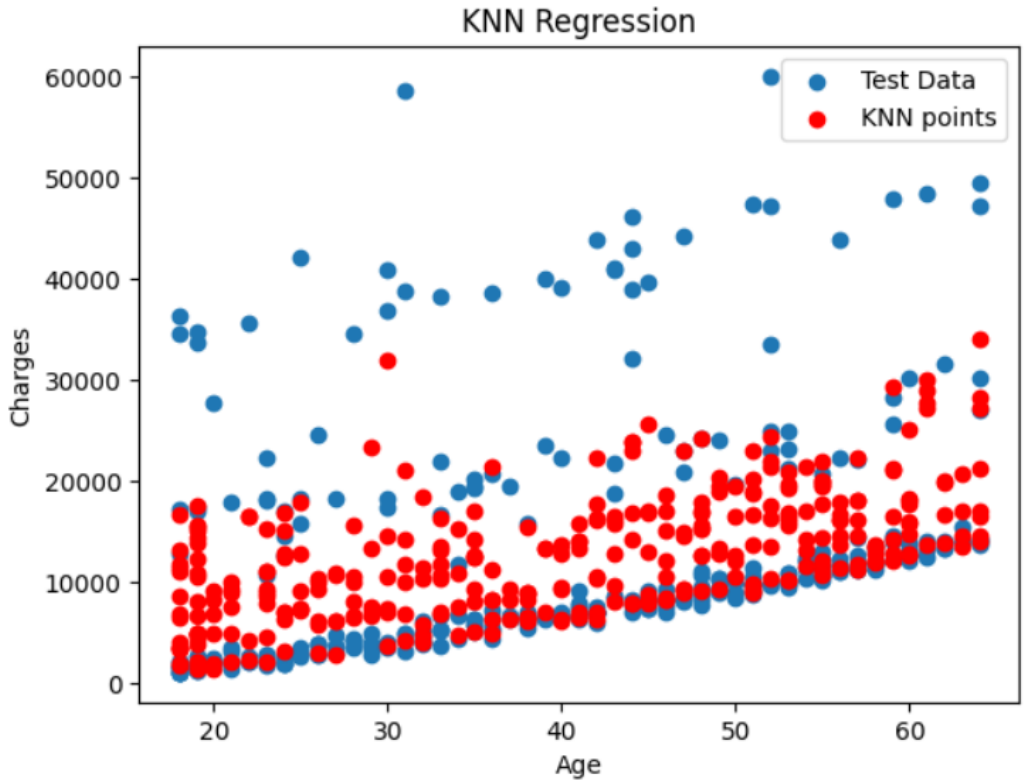
	Predicted	Actual
559	15156.859640	1646.42970
1087	18192.719010	11353.22760
1020	9134.221920	8798.59300
460	20422.064668	10381.47870
802	4921.118030	2103.08000
...	...	...
1192	12195.102240	13019.16105
628	13554.630800	11365.95200
1098	13513.176664	23045.56616
1038	2309.642870	2250.83520
936	7866.544660	32108.66282

335 rows x 2 columns

```
plt.scatter(X_test['smoker_yes'], y_test, label='Test Data')
plt.scatter(X_test['smoker_yes'], KNN_y_pred, color='red', linewidth=1, label='KNN points')
plt.xlabel('Smoker status')
plt.ylabel('Charges')
plt.title('KNN Regression')
plt.legend()
plt.show()
```




```
plt.scatter(X_test['age'], y_test, label='Test Data')
plt.scatter(X_test['age'], KNN_y_pred, color='red', linewidth=1, label='KNN points')
plt.xlabel('Age')
plt.ylabel('Charges')
plt.title('KNN Regression')
plt.legend()
plt.show()
```



```python
# try different values of n_neighbors
for i in range(1, 50):
  Knn_model = KNeighborsRegressor(n_neighbors=i)
  Knn_model.fit(X_train, y_train)
  Knn_y_pred = Knn_model.predict(X_test)
  print(f"n_neighbors = {i}, R^2 score: {r2_score(y_test, Knn_y_pred)}")
```

```
n_neighbors = 1, R^2 score: -0.5292927084594565
n_neighbors = 2, R^2 score: -0.04493687178096328
n_neighbors = 3, R^2 score: 0.10696172769329693
n_neighbors = 4, R^2 score: 0.1064600621737678
n_neighbors = 5, R^2 score: 0.1525119628149023
n_neighbors = 6, R^2 score: 0.14304292705479538
n_neighbors = 7, R^2 score: 0.15756127576172918
n_neighbors = 8, R^2 score: 0.16877137022666377
n_neighbors = 9, R^2 score: 0.17944754440291766
n_neighbors = 10, R^2 score: 0.19037622511420482
n_neighbors = 11, R^2 score: 0.19478236154586603
n_neighbors = 12, R^2 score: 0.18412092255608625
n_neighbors = 13, R^2 score: 0.16661262061619142
n_neighbors = 14, R^2 score: 0.15349363378088854
n_neighbors = 15, R^2 score: 0.14911959663557717
n_neighbors = 16, R^2 score: 0.14819915536423311
n_neighbors = 17, R^2 score: 0.14496632499271223
n_neighbors = 18, R^2 score: 0.13980223968689387
n_neighbors = 19, R^2 score: 0.14870276889288514
n_neighbors = 20, R^2 score: 0.15125636160849065
n_neighbors = 21, R^2 score: 0.14644153489863299
n_neighbors = 22, R^2 score: 0.15004786367370804
n_neighbors = 23, R^2 score: 0.15913298176270807
n_neighbors = 24, R^2 score: 0.16414740366929603
n_neighbors = 25, R^2 score: 0.17263936718050277
n_neighbors = 26, R^2 score: 0.16848115010709153
```

```python
# Different splitting ratios
split_ratios = [0.1, 0.2, 0.3, 0.4]

for ratio in split_ratios:
    Knn_X_train, Knn_X_test, Knn_y_train, Knn_y_test = train_test_split(X_1, y_1, test_size=ratio, random_state=1)
    KNN = KNeighborsRegressor(n_neighbors=5)
    KNN.fit(Knn_X_train, Knn_y_train)
    Knn_y_pred = KNN.predict(Knn_X_test)
    Knn_rmse = np.sqrt(metrics.mean_squared_error(Knn_y_test, Knn_y_pred))
    Knn_R2 = r2_score(Knn_y_test, Knn_y_pred)
    print(f"Split Ratio: {ratio}, RMSE: {Knn_rmse}, R Score: {Knn_R2}")
```

```
Split Ratio: 0.1, RMSE: 12167.606628899475, R Score: 0.06311135352583308
Split Ratio: 0.2, RMSE: 11130.460425849844, R Score: 0.17010571756030213
Split Ratio: 0.3, RMSE: 11037.968808585956, R Score: 0.14027931286970075
Split Ratio: 0.4, RMSE: 11582.532748924066, R Score: 0.06666751372539925
```

SVM REGRESSOR ALGORITHM

```
SVR_model = SVR(kernel='linear')
```

```
SVR_model.fit(X_train, y_train)
```

```
SVR(kernel='linear')
```

```
SVR_y_pred = SVR_model.predict(X_test)
SVR_y_pred
```

```
array([ 1729.84604621, 11796.66969458, 10165.12005253, 10442.82210715,
         2439.08040565,  5831.47087174, 10146.77850771, 11592.13745506,
         3844.76425765,  6755.39502335, 12636.94234792, 11502.14711724,
         7492.25202855,  8069.9629807 ,  1455.22528409,  9837.91981555,
         5428.88648487,  6687.47664552, 12619.11366078, 13136.88107045,
        10476.16208408,  5621.77668224,  8938.27816509,  9891.14711275,
         1523.18438032,  7790.6974942 ,  8796.54624545,  9464.44576054,
         5925.1043038 ,  4639.29365348, 11785.92244474,  6125.06396891,
        11714.48131925,  1938.73380766, 10380.52156331, 10730.41748158,
         4810.50884199,  3364.53602989, 12119.44640346,  9124.48783722,
         4637.70553617, 12064.13365899, 10700.24242083, 12522.10321258,
         5387.07275758, 12592.20440615,  2222.83929074,  4314.92380497,
        10254.95893147, 12804.67409777, 12848.05779333, 11752.49808019,
         3044.64308926,  9335.82175336,  1909.68994277,  5233.72325959,
```

```
svm = pd.DataFrame({'Predicted':SVR_y_pred,'Actual':y_test})
svm
```

	Predicted	Actual
559	1729.846046	1646.42970
1087	11796.669695	11353.22760
1020	10165.120053	8798.59300
460	10442.822107	10381.47870
802	2439.080406	2103.08000
...	...	...
1192	12362.268840	13019.16105
628	12021.030706	11365.95200
1098	10534.075344	23045.56616
1038	2522.186618	2250.83520
936	8826.759228	32108.66282

335 rows × 2 columns

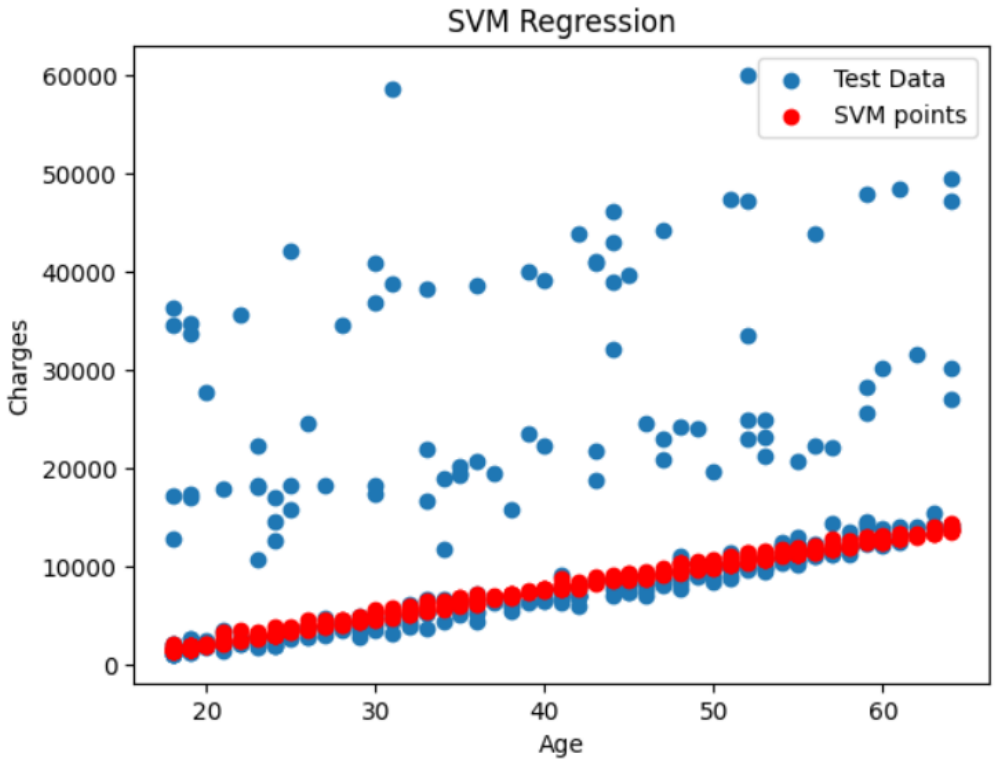
```
r2_score(y_test,SVR_y_pred)
```

```
-0.08818394289501663
```

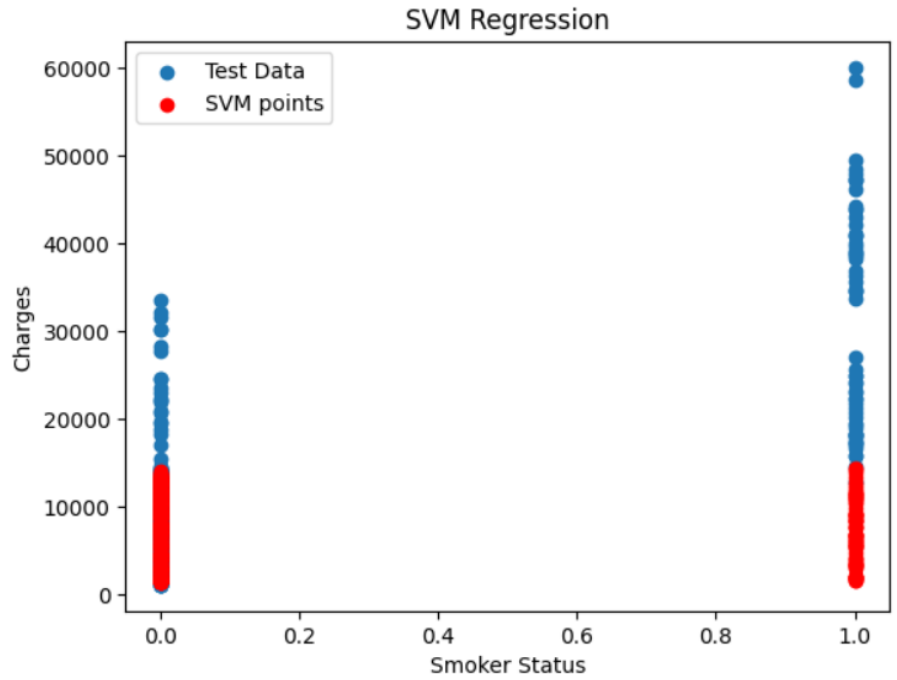
```
metrics.mean_absolute_error(y_test,SVR_y_pred)
```

```
5894.915677187849
```

```
plt.scatter(X_test['age'], y_test, label='Test Data')
plt.scatter(X_test['age'], SVR_y_pred, color='red', linewidth=1, label='SVM points')
plt.xlabel('Age')
plt.ylabel('Charges')
plt.title('SVM Regression')
plt.legend()
plt.show()
```




```
plt.scatter(X_test['smoker_yes'], y_test, label='Test Data')
plt.scatter(X_test['smoker_yes'], SVR_y_pred, color='red', linewidth=1, label='SVM points')
plt.xlabel('Smoker Status')
plt.ylabel('Charges')
plt.title('SVM Regression')
plt.legend()
plt.show()
```



```
kernels = ['linear', 'poly', 'rbf', 'sigmoid']
for kernel in kernels:
  SVR_model = SVR(kernel=kernel)
  SVR_model.fit(X_train, y_train)
  SVR_y_PRED = SVR_model.predict(X_test)
  print(f"Kernel = {kernel}, R^2 score: {r2_score(y_test, SVR_y_PRED)}")
```

```
Kernel = linear, R^2 score: -0.08818394289501663
Kernel = poly, R^2 score: -0.07666024116227921
Kernel = rbf, R^2 score: -0.09246213776984025
Kernel = sigmoid, R^2 score: -0.09380102160682369
```

```
[ ] # Different splitting ratios
split_ratios = [0.1, 0.2, 0.3, 0.4]

for ratio in split_ratios:
    SVR_X_train, SVR_X_test, SVR_y_train, SVR_y_test = train_test_split(X_1, y_1, test_size=ratio, random_state=1)
    SVR_model = SVR(kernel='linear')
    SVR_model.fit(SVR_X_train, SVR_y_train)
    Svr_y_pred = SVR_model.predict(SVR_X_test)
    SVR_rmse = np.sqrt(metrics.mean_squared_error(SVR_y_test, Svr_y_pred))
    SVR_R2 = r2_score(SVR_y_test, Svr_y_pred)
    print(f"Split Ratio: {ratio}, RMSE: {SVR_rmse}, R Score: {SVR_R2}")
```

```
Split Ratio: 0.1, RMSE: 14422.247643842617, R Score: -0.3162656611582748
Split Ratio: 0.2, RMSE: 13013.087991714845, R Score: -0.13437656431767087
Split Ratio: 0.3, RMSE: 12619.18428134492, R Score: -0.12367732495468231
Split Ratio: 0.4, RMSE: 12960.348445702371, R Score: -0.16859134294415923
```

DECISION TREE ALGORITHM

```
[ ] dt_model = DecisionTreeRegressor()
```

```
[ ] dt_model.fit(X_train, y_train)
```

```
DecisionTreeRegressor
DecisionTreeRegressor()
```

```
[ ] DT_y_pred = dt_model.predict(X_test)
```

```
DT_y_pred
```

```
array([ 1263.249  , 11842.62375,  8457.818  , 10702.6424 ,  1964.78   ,
       39983.42595,  8978.1851 , 11658.37915,  2483.736  , 20149.3229 ,
       13451.122  , 12105.32   ,  6238.298  ,  7640.3092 ,  1137.011  ,
       21232.18226,  4185.0979 ,  6746.7425 , 11576.13   , 13470.86   ,
        9140.951  , 39241.442  ,  7731.85785,  8782.469  , 21344.8467 ,
        6393.60345,  8891.1395 ,  8280.6227 , 14358.36437,  2866.091  ,
       11763.0009 ,  4746.344  , 22218.1149 , 36898.73308, 21098.55405,
       10197.7722 , 36950.2567 , 16232.847  , 24227.33724, 40974.1649 ,
       18903.49141, 12231.6136 , 10702.6424 , 11576.13   ,  4719.52405,
       13217.0945 ,  1391.5287 , 37270.1512 ,  9566.9909 , 13430.265  ,
       13143.86485, 12105.32   ,  2395.17155, 26236.57997, 16450.8947 ,
        3989.841  , 40974.1649 , 12638.195  ,  3180.5101 ,  1146.7966 ,
        3268.84665, 13143.86485, 35147.52848,  2850.68375, 13228.84695,
        8964.06055, 10600.5483 , 12044.342  ,  2710.82855, 13747.87235,
```

```
dtr = pd.DataFrame({'Predicted':DT_y_pred,'Actual':y_test})
dtr
```

	Predicted	Actual
559	1263.24900	1646.42970
1087	11842.62375	11353.22760
1020	8457.81800	8798.59300
460	10702.64240	10381.47870
802	1964.78000	2103.08000
...	...	...
1192	11554.22360	13019.16105
628	20709.02034	11365.95200
1098	10797.33620	23045.56616
1038	2741.94800	2250.83520
936	8310.83915	32108.66282

335 rows × 2 columns

```
[ ] r2_score(y_test,DT_y_pred)
```

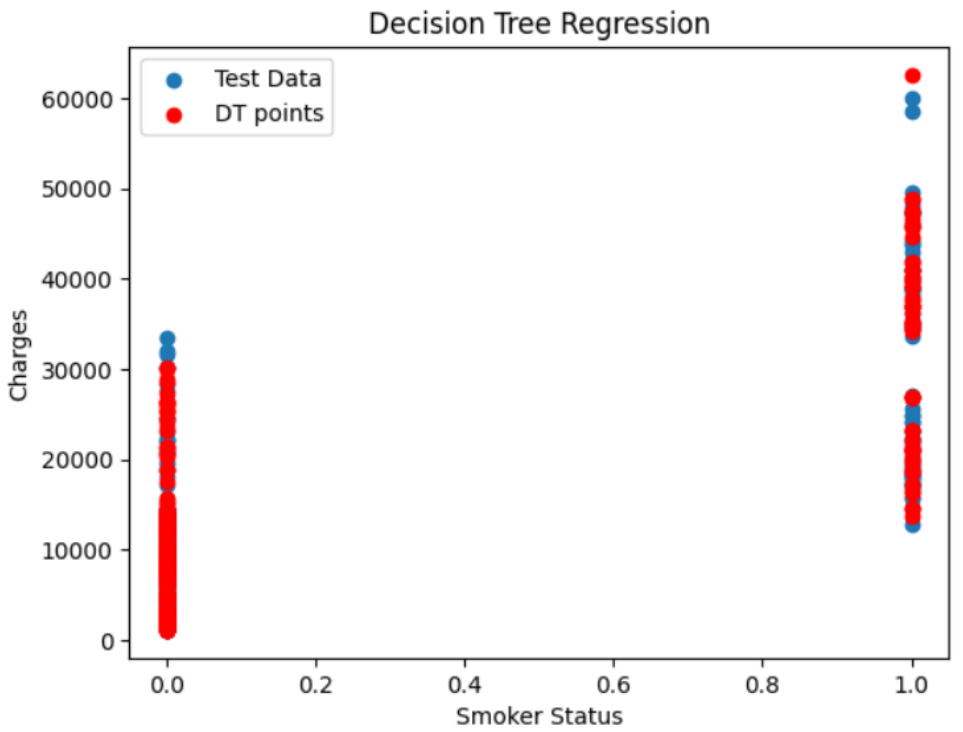
```
0.6604425404213328
```

```
[ ] metrics.mean_absolute_error(y_test,DT_y_pred)
```

```
3300.103573044776
```

```
plt.scatter(X_test['age'], y_test, label='Test Data')
plt.scatter(X_test['age'], DT_y_pred, color='red', linewidth=1, label='DT points')
plt.xlabel('Age')
plt.ylabel('Charges')
plt.title('Decision Tree Regression')
plt.legend()
plt.show()
```


```
plt.scatter(X_test['smoker_yes'], y_test, label='Test Data')
plt.scatter(X_test['smoker_yes'], DT_y_pred, color='red', linewidth=1, label='DT points')
plt.xlabel('Smoker Status')
plt.ylabel('Charges')
plt.title('Decision Tree Regression')
plt.legend()
plt.show()
```



RANDOM FOREST REGRESSOR ALGORITHM

```
[ ] rf = RandomForestRegressor()
```

```
[ ] rf.fit(X_train, y_train)
```

```
RandomForestRegressor
RandomForestRegressor()
```

```
RF_y_pred = rf.predict(X_test)
RF_y_pred
```

```
array([ 1879.7733835 , 12192.0872003 ,  9187.9298866 , 11059.6690133 ,
        2067.747822  , 40011.1553402 , 10730.3512142 , 11654.5016325 ,
        3124.017632  , 20116.0451127 , 17776.9283678 , 13884.1207318 ,
        6784.18084   ,  7284.7830483 ,  2230.240814  , 12009.9675988 ,
        6251.5107336 ,  7333.5325532 , 16216.2556748 , 13675.6226848 ,
        9515.019639  , 41316.3043612 , 12465.8629475 , 13612.6249612 ,
       14833.46472526,  6719.3685565 ,  8575.2307821 ,  9996.6402223 ,
        7885.3254153 ,  3672.877692  , 12421.8567794 ,  7441.7778876 ,
       25166.3759225 , 35179.0606756 , 26129.3363955 , 14493.7233588 ,
       39491.4926932 , 17321.1810833 , 16032.5383192 , 45550.9190387 ,
        6603.9077536 , 11925.678125  , 11394.8668511 , 18621.8795525 ,
        5540.9708351 , 13003.3434969 ,  1818.417561  , 37666.5065036 ,
       10981.5197458 , 16448.0137183 , 17712.6236262 , 15110.1295753 ,
        5242.8023242 , 11756.4363891 , 16562.94022   ,  6417.8292161 ,
```

```
rfr = pd.DataFrame({'Predicted':RF_y_pred,'Actual':y_test})
rfr
```

	Predicted	Actual
559	1879.773384	1646.42970
1087	12192.087200	11353.22760
1020	9187.929887	8798.59300
460	11059.669013	10381.47870
802	2067.747822	2103.08000
...	...	...
1192	12573.472090	13019.16105
628	15104.288253	11365.95200
1098	10654.374207	23045.56616
1038	2319.126492	2250.83520
936	8047.399417	32108.66282

335 rows × 2 columns

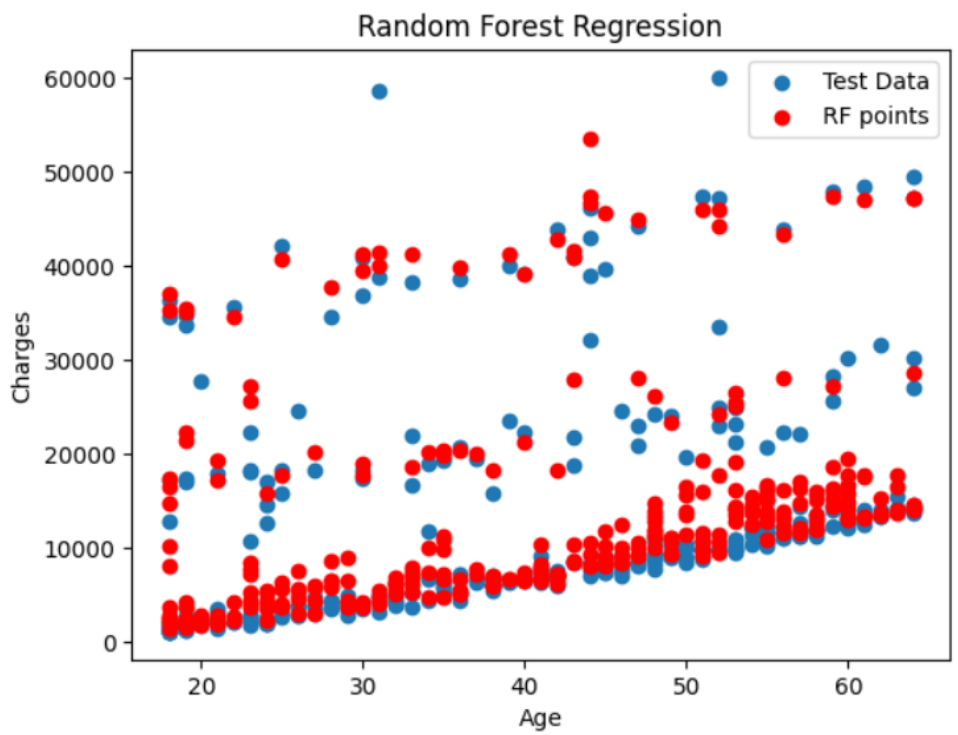
```
[ ] r2_score(y_test,RF_y_pred)
```

```
0.8196146458965754
```

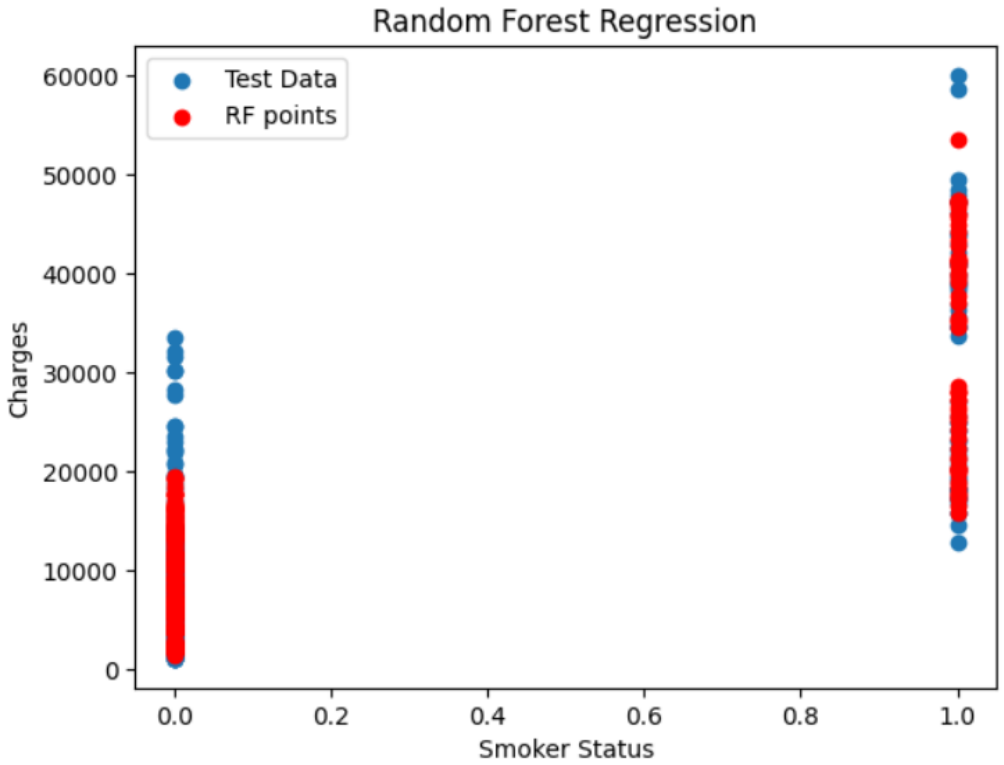
```
[ ] metrics.mean_absolute_error(y_test,RF_y_pred)
```

```
2773.4246761231348
```

```
plt.scatter(X_test['age'], y_test, label='Test Data')
plt.scatter(X_test['age'], RF_y_pred, color='red', linewidth=1, label='RF points')
plt.xlabel('Age')
plt.ylabel('Charges')
plt.title('Random Forest Regression')
plt.legend()
plt.show()
```




```
plt.scatter(X_test['smoker_yes'], y_test, label='Test Data')
plt.scatter(X_test['smoker_yes'], RF_y_pred, color='red', linewidth=1, label='RF points')
plt.xlabel('Smoker Status')
plt.ylabel('Charges')
plt.title('Random Forest Regression')
plt.legend()
plt.show()
```



BAGGING ALGORITHM

```
base_model = DecisionTreeRegressor(max_features = "sqrt")
model_BA = BaggingRegressor(base_model, n_estimators = 100, random_state = 1)
BA = model_BA.fit(X_train, y_train)
BA_y_pred = model_BA.predict(X_test)
BA_y_pred
```

```
array([26603.3480891 , 20526.8853471 ,  2768.6005382 ,  8237.234579  ,
        3915.5038239 , 12868.7766712 ,  4525.89989   ,  4376.993925  ,
        2287.3415049 , 24043.9137426 ,  4584.763755  ,  3688.32236779,
        8138.2987072 , 16236.3013329 , 10992.7683225 ,  2987.0526976 ,
       12771.5025356 , 13003.9262541 ,  6655.5409532 , 10962.2028675 ,
       39496.4819308 , 11492.9163202 , 13400.5126598 , 21558.2936079 ,
       37079.7343483 ,  8577.0509644 , 35519.0196865 ,  2332.5925146 ,
        7310.8725904 , 37278.8455169 , 13109.5665773 ,  2024.1466945 ,
        2250.2093869 , 10013.414094  , 10304.1701    , 10710.8370513 ,
       13811.1492793 ,  6197.9173178 , 19621.7030817 , 14648.1027579 ,
       16573.3166138 , 14025.9615113 ,  2469.95503089,  8710.8708173 ,
        2525.0139582 ,  7479.7767516 ,  9753.0911924 ,  5228.6969139 ,
       14293.1964809 , 12939.5551522 ,  8764.863942  ,  9119.4703072 ,
       11647.6373354 ,  9866.1603941 , 10610.2638438 ,  3642.0415763 ,
        7823.9410332 , 41392.4460618 , 14087.5424122 ,  4154.6885503 ,
        8262.3172275 ,  3725.7603012 , 37776.0740482 , 11768.0678229 ,
       12569.6570693 ,  5168.4325615 ,  7656.4599859 ,  5163.8666284 ,
       26908.4575008 , 27649.1272991 ,  4440.7914594 ,  7293.2560813 ,
```

```
BaR = pd.DataFrame({'Predicted':BA_y_pred,'Actual':y_test})
BaR
```

	Predicted	Actual
559	3483.574265	1646.42970
1087	11432.888602	11353.22760
1020	10137.824735	8798.59300
460	12654.769514	10381.47870
802	3582.366873	2103.08000
...	...	...
1192	12877.348316	13019.16105
628	16709.911879	11365.95200
1098	12051.339358	23045.56616
1038	3314.514580	2250.83520
936	10393.345828	32108.66282

335 rows × 2 columns

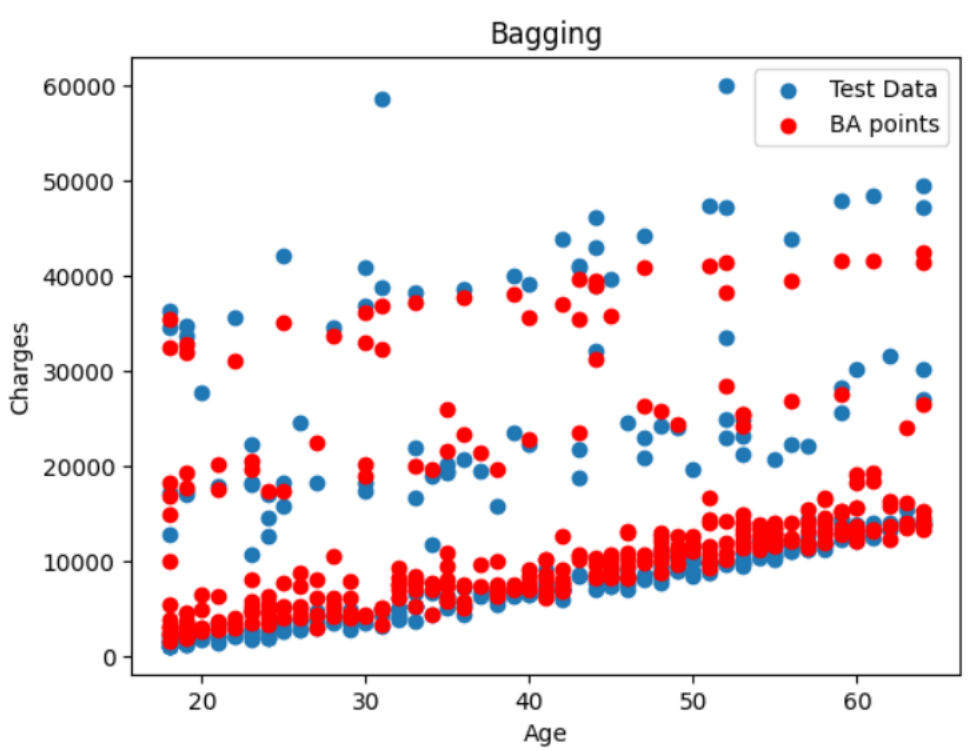
```
[ ] r2_score(y_test,BA_y_pred)
```

```
0.8181018584358947
```

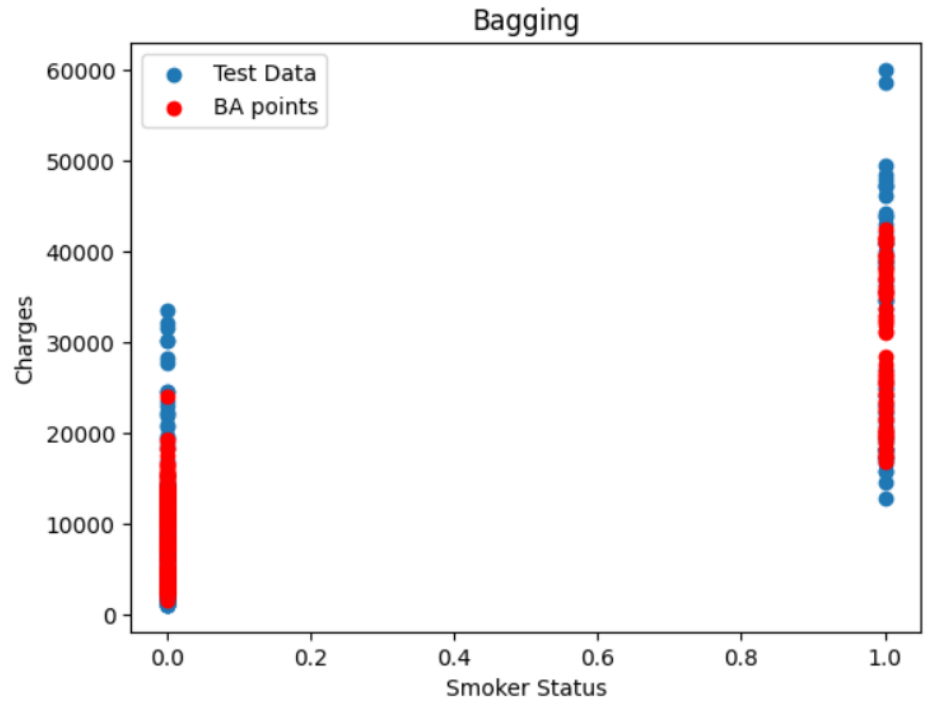
```
[ ] metrics.mean_absolute_error(y_test,BA_y_pred)
```

```
2980.0407953655217
```

```
[ ]  plt.scatter(X_test['age'], y_test, label='Test Data')
     plt.scatter(X_test['age'], BA_y_pred, color='red', linewidth=1, label='BA points')
     plt.xlabel('Age')
     plt.ylabel('Charges')
     plt.title('Bagging')
     plt.legend()
     plt.show()
```




```
plt.scatter(X_test['smoker_yes'], y_test, label='Test Data')
plt.scatter(X_test['smoker_yes'], BA_y_pred, color='red', linewidth=1, label='BA points')
plt.xlabel('Smoker Status')
plt.ylabel('Charges')
plt.title('Bagging')
plt.legend()
plt.show()
```



BOOSTING ALGORITHM

```
[ ] boost_model1 = xgb.XGBRegressor()
    boost_model2 = xgb.XGBRegressor(n_estimators=100, max_depth=8, learning_rate=0.1, subsample=0.5)

    boost_train_model1 = boost_model1.fit(X_train, y_train)
    boost_train_model2 = boost_model2.fit(X_train, y_train)

    boost_pred1 = boost_train_model1.predict(X_test)
    boost_pred2 = boost_train_model2.predict(X_test)

    print(f'Model 1(1) R Score: {(r2_score(y_test, boost_pred1))}')
    print(f'Model 1(2) R Score: {(r2_score(y_test, boost_pred2))}')
```

```
Model 1(1) R Score: 0.8033076042141283
Model 1(2) R Score: 0.8157395961121844
```

Boosting after removing Multicolinearity:

```
[ ] boost_model1_1 = xgb.XGBRegressor()
    boost_model2_1 = xgb.XGBRegressor(n_estimators=100, max_depth=8, learning_rate=0.1, subsample=0.5)

    boost_train_model1_1 = boost_model1_1.fit(X_train_1, y_train_1)
    boost_train_model2_1 = boost_model2_1.fit(X_train_1, y_train_1)

    boost_pred1_1 = boost_train_model1_1.predict(X_test_1)
    boost_pred2_1 = boost_train_model2_1.predict(X_test_1)

    print(f'Model 2(1) R Score: {(r2_score(y_test_1, boost_pred1_1))}')
    print(f'Model 2(2) R Score: {(r2_score(y_test_1, boost_pred2_1))}')
```

```
Model 2(1) R Score: 0.4987989900266372
Model 2(2) R Score: 0.5346358344835345
```

```
#R Scores:

print(f'Model 1(1) R Score: {(r2_score(y_test, boost_pred1))}')
print(f'Model 1(2) R Score: {(r2_score(y_test, boost_pred2))}')
print(f'Model 2(1) R Score: {(r2_score(y_test_1, boost_pred1_1))}')
print(f'Model 2(2) R Score: {(r2_score(y_test_1, boost_pred2_1))}')
print(f'Model 3(1) R Score: {(r2_score(y_test_2, boost_pred1_2))}')
print(f'Model 3(2) R Score: {(r2_score(y_test_2, boost_pred2_2))}')
print(f'Model 4(1) R Score: {(r2_score(y_test_3, boost_pred1_3))}')
print(f'Model 4(2) R Score: {(r2_score(y_test_3, boost_pred2_3))}')
```

```
Model 1(1) R Score: 0.8033076042141283
Model 1(2) R Score: 0.8157395961121844
Model 2(1) R Score: 0.4987989900266372
Model 2(2) R Score: 0.5346358344835345
Model 3(1) R Score: 0.7949992609992695
Model 3(2) R Score: 0.8107420860814198
Model 4(1) R Score: 0.5125069329230832
Model 4(2) R Score: 0.5494500947131551
```